ON RECOMMENDATIONS IN HETEROGENEOUS SOCIAL MEDIA NETWORKS

BY

MIN-HSUAN TSAI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Doctoral Committee:

Professor Thomas S. Huang, Chair
Dr. Charu Aggarwal, IBM T.J. Watson Research Center
Professor Mark Hasegawa-Johnson
Professor Zhi-Pei Liang

ABSTRACT

In this dissertation, we study the problem of social media recommendations with a heavy emphasis on exploiting social, content and contextual information. The problem of recommendation analysis and collaborative filtering has been widely studied in the literature because of its numerous applications to a wide variety of scenarios. Many social media sites such as Flickr or YouTube contain multimedia objects, which occur in the context of an extensive amount of content information such as tags, image content, in addition to the user preferences for the different objects. Thus, there is a plethora of *heterogeneous, content, linkage and preference information* in a social media network, which can be used in order to make effective recommendations in such networks. In this dissertation, we will study the problem of making recommendations in such complex multimedia networks with the use of such information. While our approach is developed and evaluated for the case of the Flickr image network, the broad principles are applicable to any kind of multimedia network such as a music or video site. To ensure the efficiency and scalability, we further extend our approach to incorporate the latent factor model so that our approach can be very useful for making personalized content recommendations in large and heterogeneous social media networks. This dissertation also studies a variety of recommendation scenarios, including context-specific recommendations which are made based on some kinds of content the user specifies, cold start recommendations that utilizes the social relations to make recommendations when a new user gets on board, and preference drifting to realize the long-term change of users' tastes. We present experimental results illustrating the effectiveness and efficiency of our approaches.

To my family, for their love and support

ACKNOWLEDGMENTS

I wish to express my utmost appreciation and gratitude to my advisor, Prof. Thomas S. Huang, for his valuable guidance, supervision and encouragement throughout the course of my graduate study. I owe sincere and deepest gratitude to Dr. Charu Aggarwal from IBM T. J. Watson Research Center, for his constant inspiration, enthusiastic suggestions and continuous encouragement. Special thanks go to Prof. Mark Hasegawa-Johnson and Prof. Zhi-Pei Liang for serving on my dissertation committee and for providing critical suggestions and valuable advice.

I am grateful to my collaborators: Dr. Yihong Gong, Prof. Tong Zhang, Dr. Huazhong Ning and Dr. Jinjun Wang. It has been my honor to work with them and learn from them. I would also like to thank all the members of the Image Formation and Processing (IFP) group at Beckman Institute, especially Liangliang Cao, Zhen Li, Xianming Liu, Huazhong Ning, Guo-Jun Qi, Shen-Fu Tsai and Xi Zhou (in alphabetical order). Thanks also go to many other friends at University of Illinois at Urbana-Champaign. I appreciate their help and support to make my PhD life colorful and exciting.

I also give special thanks to my host family, Grandma Marge, for her love, care, encouragement throughout the years. Last but not least, I am most indebted to my dear family: my parents, parents-in-law, my wife Yi-Shuan and my daughter Alicia, for their unconditional spiritual support, patience, encouragement and company throughout several challenging times and crucial moments during my studies. Their long-lasting support has made this dissertation possible. This dissertation is dedicated to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Social media platforms such as Flickr, YouTube, Facebook, Twitter or other music sharing sites serve as a rich platform for sharing different kinds of content between users. With the rapid growth of these sharing platforms which provide incredibly easy ways to create and upload media objects from computers and/or mobile devices, the users are extremely overwhelmed by the media content shared and provided from the social media sites. Therefore it is a crucial task for such a platform to allow the user to find relevant content, to recommend useful content to the different users, and also to determine the most relevant users for the content-objects and vice versa. Such tasks are often categorized as the *information filtering*, or more popularly, the *recommendation* problem.

In addition to the media content hosted in these media repositories, such platforms also provide a rich level of information in terms of the user linkage information to media objects (e.g., images, videos, music songs,...), tags or other comments which are contributed by the users. It is reasonable to assume that the media content, users and other social cues such as tags and comments are often related to one another. Such relationship-rich and content-intensive cues of different kinds indeed form a *heterogeneous social media network* that has useful information for improving the effectiveness of the information filtering techniques.

While some social media platforms provide users the ability to rate their interest in the underlying content explicitly, with the use of a *"star rating system"*, such information is extremely sparse in many social media networks since a user may exhibit a preference for only a small fraction of the media objects and vice versa. Furthermore, most of the preference information is *implicit*, in that explicit user ratings are often not available (except as a binary *like* tag), and the interest of the users for a given media object may only be inferred from their actions, such as a *like* link, or other

contribution linkages between users or images. While it is possible to perform recommendations purely with the use of collaborative filtering techniques such as those discussed in [1, 2, 3], such an approach ignores the much richer level of information which is available in terms of user comments, tags, implicit linkages, and the actual media content. For example, if a user has exhibited an interest in images which are tagged by the keyword "northern lights", it is likely that the user will also be interested in other images corresponding to this same subject. Furthermore, the similarity in *media content* between the different media objects can also play a role in the information filtering process. This creates unprecedented challenges, because the combination of media content, textual information and linkage information (in the form of user preferences and other tagging or user-contribution links) provides a rather *heterogeneous* scenario for the recommendation process.

A social media network can be considered a network of media objects, text comments/tags and actors which contribute and share these media objects. A social media network is *heterogenous*: it consists of the nodes of different kinds and there may also be linkages of different kinds in the network. For example, the contribution of an image to a social media site such as Flickr constitutes a linkage between that actor and that image. A comment, tag or "like" label by a user in any social network also constitutes a link between the user and an image in the network. Such a comment also constitutes a link between the underlying text and image node. Thus, a social network may be modeled as a network of images, text, and users with heterogeneous links of different kinds between different nodes. In fact, such a network can be represented as a graph structure with nodes of different types and links representing the relationships between them. In the context of a social media network such as the Flickr network, the three kinds of nodes in a social network are *actor nodes*, *text nodes* and *image nodes*.

- The *actor* nodes correspond to users who may either contribute, comment on, or tag the images in the network. Such an actor may be linked to either text nodes corresponding to their comments, tags, wall posts, or to image nodes depending upon their image contributions, or sharing behavior. The different kinds of links may refer to the relationship between the actor and the underlying text.

The actors may also be directly connected to one another with the use of friendship or membership links.

- The image nodes represent the set of images in the social network which may be recommended to different users. Such images are linked to the nodes corresponding to their surrounding text, the tags or comments by users, and wall posts depending upon the association between the underlying text and the image. The image nodes are also linked to the actors based on contribution, or sharing behavior of likeability flags. Furthermore, in order to incorporate the effect of *content-based* similarity, images may also be linked to one another based on a measure of visual similarity.

- The text nodes represent the copious amount of text available in any social or web network. The text nodes may be connected to either actor nodes or image nodes. For example, a comment by a user on an image could be considered a text node which links to both that user and the corresponding image.

The social media network structure and content may contain rich information which can be leveraged for improving the effectiveness of the recommendation process. An example is the *like* tag [4], which provides useful information about the preferences of the users for social media objects. An important desiderata for such applications is to be able to focus the recommended results to a particular subject, as may be specified by a particular user. For example, in the context of an image media network, it may be possible to focus the recommendation results in terms of different keywords or image targets. This leads to recommendations with varying levels of specificity, as follows:

- For a given user, determine the highest ranked image recommendation.

- For a given user, determine the highest ranked image recommendations related to the keyword "northern lights".

- For a given user, a target image, and the keyword "northern lights" determine the highest ranked image recommendations related to the target image and the keyword "northern lights".

- For a given group of users, determine the highest ranked image recommendation.
- For a given image keyword "northern lights", determine the most likely users to be interested in that subject.

The first four kinds of scenarios are useful for making *context-specific recommendations* to a specific user or a group of users, whereas the last one may be useful for determining individuals with interests in specific kinds of content. We note that the nature of the recommendation contexts are quite complex, in such scenarios, and require knowledge of the attribute and linkage information related to the objects. Clearly, straightforward collaborative filtering methods [1, 2, 3] cannot be used directly in such scenarios because of the complexity of finding recommendations related to specific kinds of content.

The rest of the dissertation is organized as follows. In Chapter 2 we first discuss the related work. In Chapter 3 we present the preliminaries for the recommendation problem and introduce the heterogeneous network model (HNM) that extends the Bayesian regularization model for heterogeneous networks. Chapter 4 introduces two hybrid models that combine the latent factor model with our proposed HNM regularization from two different aspects. Then in Chapter 5 we discuss various real-world recommendation scenarios with the proposed models. Finally in Chapter 6 we make conclusions and explore some future directions.

CHAPTER 2

RELATED WORK

The problem of recommendation analysis has been widely studied in the literature [5, 6]. Recommendations can be performed with an analysis of explicit user-specified ratings [1, 7, 8, 9, 10] or implicit one-class user click behavior/like specifications [11, 12, 13], a process which is also known as *collaborative filtering*. The traditional challenge of such methods is the extreme sparsity of user-specified feedback. Another class of methods, known as *content-based techniques* [14, 15], use the keyword content-information in either user profiles or user-behavior in order to perform the recommendations. Some recent work has also been designed to perform the recommendations with a combination of collaborative and content information [16, 17, 18, 19]. Some recommendation methods have also been designed for specific kinds of multimedia data such as music [16] and video [20] data, though these methods are not designed in the context of social networks.

More recently, the use of tags and/or other attributes associated with the items has been studied in order to improve the effectiveness of the recommendation process [21, 22, 23, 24]. In particular, [21] introduces a homogeneous network model to incorporate social tagging information for the recommendation process. Singh and Gordon [24] propose a collective matrix factorization model to simultaneously learn latent factors from several relational matrices related to the items. However, none of these previous works are designed for complex heterogeneous networks containing different kinds of content and/or linkage information.

It has been observed in recent work that social neighbors, which are based on links, often contain a rich amount of information [25], which can be leveraged for improving the quality of the recommendation. Examples of the social relationships include but are not limited to friendships [21, 26], memberships [27, 28] and trust relationships between users [29, 30]. This

general principle has also been applied in the context of social network structure analysis [31, 32, 33] in order to establish effective recommendations; nevertheless these methods use little or no content information during the recommendation process.

Another challenge for the recommender systems is the cold-start user problem when the user has no preference history (e.g., when a new user just gets on board). Some recent work [26, 34, 35] has proposed to establish the recommendation based on the users' social relations. However, as the model training is not incremental, the cost of retraining new models for new users is expensive.

Furthermore, some recent methods [21, 36, 37, 38] use a limited amount of heterogeneous content and network structure information for recommendations. For example, Gu, Zhou and Ding [37] propose to incorporate user and item information into the weighted non-negative matrix factorization method via graph regularization. It is worth noting our method is quite different than that in [37] in the sense that their method only applies the regularization within each domain while our proposed method also considers the regularization across domains. Later, Yu et al. [38] extend the work to incorporate item similarities based on pre-defined cross-domain relationships which are called *meta-paths*. However, this method relies on the prior knowledge to manually identify meta-paths that may not be complete enough to capture all the useful relationships. We will have more close comparisons with these methods in the next few chapters.

Finally, our approach also allows for *context-specific* recommendations, where a user can determine recommendations targeted to specific kinds of content. Such broad functionality is not available in the current recommender systems. We note that this new scenario is different from the recently popular context-aware recommendation [39, 40, 41] where the "context" is referred to as the information that is associated with both the user and the item at the same time, such as the location or the time where the user-item interactions happen.

CHAPTER 3

HETEROGENEOUS NETWORK MODEL

In this section, we define a heterogeneous network model for social media networks, which expressively utilizes multiple types of objects, linkages and their importance in the recommendation process. We further note that the general principles of our approach can easily be extended to any kind of heterogeneous network and any kind of complex multimedia recommendations, though the focus of this proposal is on networks of images, text and actors.

Since the social network recommendation process will use nodes with different kinds of data (text content, image content and users), we will first define a heterogeneous network model with different node *types*. We denote the set of types of nodes by $\mathcal{D}$ with the corresponding cardinality $|\mathcal{D}|$ denoted by N ($N \geq 1$). We define a graph $G = \langle V, E, C \rangle$ with N types of nodes in V, the edges (or relations) between the nodes denoted by E, and the content sets attached to the different nodes by C. We note that C contains a data record for each node in V, which may be either text, image or a user identifier, depending upon the node type. Both the social network structure and the heterogeneous content such as a text and visual similarity will play a key role in the recommendation process. We refer to such a heterogeneous network of nodes, edges and weights as an information network. Note that a heterogeneous network has more than one type of node ($N \geq 2$), whereas a homogeneous network has only one type of node ($N = 1$).

In the case of heterogeneous networks, the graph G can be decomposed into the N homogeneous subgraphs $\{G^{T_i}\}_{i=1}^{N}$ whose nodes are of the same type and the inter-subgraph heterogeneous links E^H, as illustrated in Figure 3.1. Specifically, we can decompose the sets of nodes, links and the content of G into the constituent homogeneous components and intra-component edges

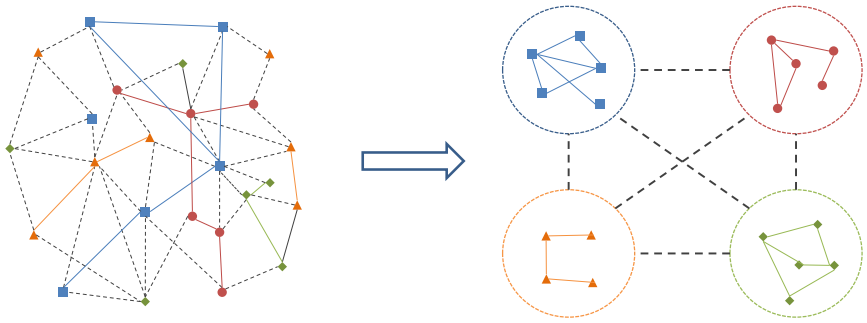


Figure 3.1: Decomposition of heterogeneous network into homogeneous sub-networks connected by heterogeneous links (dashed black lines)

as follows:

$$V(G) = V(G^{T_1}) \cup V(G^{T_2}) \cup \ldots \cup V(G^{T_N})$$
$$E(G) = E(G^{T_1}) \cup E(G^{T_2}) \cup \ldots \cup E(G^{T_N}) \cup E^H$$
$$C(G) = C(G^{T_1}) \cup C(G^{T_2}) \cup \ldots \cup C(G^{T_N})$$

Here E^H is the set of heterogeneous links between nodes from different sub-graphs.

We assume that the user preference information in the network to objects is encoded by *preference links* between users and objects. Specifically, a link between a user and an object implies an interest between the user and that object. On the other hand, the absence of a link does not necessarily imply otherwise. Such *preference links* are extremely common in social media networks. For example, in most social media networks such as Flickr, a mechanism is provided for users to specify their interests in the different objects explicitly. This can be considered analogous to (or rather a binary version of) ratings, which are used so commonly in many recommendation applications. As with all collaborative applications, such information is extremely sparse, which makes the problem rather challenging. We note that in addition to the explicit preference links, our approach will use *all available information* such as other kinds of social links, textual content and visual similarity to construct *generalized links* for the recommendation process. As we will see later in Section 3.4, such textual content and visual similarity are also encoded as *content-based links*, an approach which allows

for the use of content-information in the recommendation process. Thus, our *heterogeneous network* clearly contains nodes and links of many different kinds in different domains, which allows for great expressivity, but also causes a challenge in the learning process.

3.1 Recommendations via Bayesian Regularization over Heterogeneous Networks

As mentioned earlier, our recommendation learning problem allows the use of a variety of different contexts, in which image or keyword contexts may be added to the user identity in order to provide personalized recommendations. Thus, a "recommendation query" may contain several components including the user identity and other content-based components. The recommendation problem can be formalized as a *learning problem*, in which we start off with an (easily constructed) initial recommendation *target vector* y, based on a naive similarity of the nodes in the network to the user-specified query. We assume that the target vector y contains one value for each node in the network. As we will discuss later in Section 3.2, this vector y serves as our first (rough) estimate of the similarity of the different nodes in the network to the target query. A variety of node content and social linkage similarity (of nodes to the recommendation query) will be used in order to construct this vector, a discussion, which we will defer to Section 3.4.

While such a recommendation target vector y is easy to construct, it cannot be directly used as a robust recommendation score, because it uses a relatively straightforward and myopic view of how the content and structure of the network relates to the query. This provides, at best, a noisy and incomplete estimate of the goals of the recommendation query. Therefore, in order to generate the true recommendation scores for the nodes in the network, we use a probabilistic approach to explicitly model the noise, and incorporate more information about the structure of the underlying network into the recommendation score. We assume that the observed data y is generated by random sampling of a *smooth* recommendation function f on the graph, in which values on adjacent nodes vary from one another in a gentle way. We will see later in Section 3.2 and Section 3.3 that the smoothness property of the function f plays a critical role in using the network link structure for

knowledge propagation. As in the case of y, the function f maps from each node in the network to a value.

Our goal is to "recover" an estimate of f from the observed data y, which is a partial structural and content representation of the goals of the recommendation query. We note that the recovered f provides good estimations of the recommendation scores for each node and can be used to provide a top-N list of items for each user. With this setting, we can apply a Bayesian regularization framework where the prior distribution of f and the noise model ϵ are regularized to obtain a maximum a posteriori (MAP) estimation of f. We note that this regularization on the prior distribution of f corresponds to imposing a smoothness condition on f based on the *generalized link structure* of the social network. The regularization framework derives its inspiration from [42], in which the key insight is that the function f to be learned satisfies a *smoothness property* in which the values on adjacent nodes change from one another in a smooth way. This provides an effective way for converting the noisy a priori knowledge in the recommendation target vector y into a more coherent recommendation score f, with a structural propagation mechanism.

In this section, we aim to extend the classical regularization framework for homogeneous networks to heterogeneous networks by proposing a *heterogeneous smoothness function* that allows the values on nodes in different domains to be transferred in a smooth way. This framework recognizes the fact that the values on two edge-adjacent nodes in different domains may show varying levels of propagation behavior. This corresponds to different levels of importance for different kinds of edges in the propagative learning process, a fact which translates mathematically into a *heterogeneous smoothness function*. We will first start with a discussion of how the Bayesian regularization model may be applied to the recommendation problem on a homogeneous network. Then, we will discuss its generalization to the heterogeneous case.

3.2 Bayesian Regularization Model for Homogeneous Networks

We will first start with some preliminaries, which are needed for further development of the required mathematical machinery in this proposal. Given a graph $G = \langle V, E, C \rangle$, let $\mathcal{H}(G)$ denote the Hilbert space of real-valued functions endowed with the usual inner product such that for any two functions $\phi, \varphi \in \mathcal{H}(G)$

$$\langle \phi, \varphi \rangle = \sum_{v \in V(G)} \phi(v)\varphi(v) \tag{3.1}$$

Conceptually, in the context of a network representation, our recommendation algorithm recommends image nodes for particular actor nodes and content attributes, which provide context for the recommendation. This problem can also be simplified to a homogeneous network, where we recommend similar nodes of a particular type, based on queries of the same type. Let us first discuss this simplified scenario. We then define a function $f : V \to \mathbb{R}$ in $\mathcal{H}(G)$ which assigns each node in the network a recommendation score for different users and contexts. We refer to this function f as a *recommendation score* function. We also define another function $y : V \to \mathbb{R}$ in $\mathcal{H}(V)$ as the *recommendation target* function which measures the initial estimate of similarity between the recommendation query and different nodes of the graph. This target y may be very accurate for some of the nodes (such as the social node corresponding to the recommendation subject), but may not be effectively estimated for nodes which are indirectly or structurally connected to such nodes. Therefore, for node-specific noise ϵ_v in node v, we have:

$$y(v) = f(v) + \epsilon_v, \quad \forall v \in V \tag{3.2}$$

Note that both f and y can be equivalently expressed as column vectors where $f = (f(v_1), f(v_2), \cdots, f(v_{|V|}))^\mathsf{T}$ and $y = (y(v_1), y(v_2), \cdots, y(v_{|V|}))^\mathsf{T}$, we will treat them as functions and vectors interchangeably in this proposal.

Our goal is to learn the recommendation score for different multimedia object nodes in the network with the use of a Bayesian approach. We therefore denote $p(f)$ as the a priori probability of f to embody the a priori

knowledge of the recommendation score and $p(y|f)$ as the conditional probability of the similarity values given f, which is used to model the intrinsic noise from the data. With the Bayes rule, one can obtain the maximum a posteriori (MAP) probability estimate of f as follows:

$$\begin{aligned}\hat{f} &= \arg\max \log p(f|y) \\ &= \arg\max \left(\log p(y|f) + \log p(f)\right)\end{aligned} \tag{3.3}$$

It is this posterior probability estimate of f that we are trying to compute. If the noise ϵ_v is normally distributed with variance μ, then the conditional probability $p(y|f)$ can be computed as follows:

$$p(y|f) = \frac{1}{Z_c} \exp\left(-\frac{\|y-f\|^2}{2\mu}\right) \tag{3.4}$$

Here Z_c is the normalization constant for the conditional probability, so that the probability values sum to 1. The a priori knowledge of the recommendation scores of a network is usually set to a smoothness enforcement over the neighboring vertices such that the closer adjacent neighbors would have similar scores. Specifically, the a priori probability $p(f)$ is defined as the exponential function over f.

$$p(f) = \frac{1}{Z_p} \exp\left(-\frac{\Omega(f)}{\sigma}\right) \tag{3.5}$$

Here, Z_p is another normalization constant, which ensures that $p(f)$ is a probability. The function $\Omega(f)$ is the smoothness functional for the recommendation score function f. One natural way for measuring the smoothness is to measure the local variations of the values of the function at each node along the adjacent edges [42].

$$\begin{aligned}\Omega(f) &= \frac{1}{2} \sum_{v \in V(G)} \|\nabla_v f(v)\|^2 \\ &= \frac{1}{2} \sum_{v \in V(G)} \sum_{e \vdash v} \left(\frac{\partial f}{\partial e}\bigg|_v\right)^2 \\ &= f' \Delta f\end{aligned} \tag{3.6}$$

The last equality is due to the definition of the edge derivation function f

and the discrete Laplace operator Δ (see Zhou and Scholkopf [42] for details). We use a subscript L to denote that the smoothness measure $\Omega_L(f) = f'\Delta f$ is based on the Laplacian operator.

It is worth noting that this kind of smoothness functional ensures that large variations of the values of f along an edge are penalized, because of their contributions to $\Omega_L(f)$, and the latter's subsequent contribution to the optimization formulation of Eq. (3.3). This is important in ensuring that the network structure plays a critical role in the knowledge propagation process.

By substituting probabilities Eqs. (3.4) and (3.5) into Eq. (3.3) we obtain the MAP estimate of f as follows:

$$\begin{aligned}\hat{f} &= \arg\max\left(\log p(y|f) + \log p(f)\right) \\ &= \arg\min\left(\frac{\lambda}{2}\|y - f\|^2 + \Omega_L(f)\right) \\ &= \arg\min\left(\frac{\lambda}{2}\|y - f\|^2 + f'\Delta f\right) \end{aligned} \tag{3.7}$$

Here $\lambda = \frac{\sigma}{\mu}$ is a positive number which acts as a balancing parameter between the two terms, which correspond to the noise and smoothness criteria respectively. It is worth noting that the first term in the MAP estimation that comes from the conditional probability can be interpreted as the consistency regularization between recommendation target and score values in the graph. On the other hand, the second term that comes from the prior probability can be viewed as a smoothness regularization of the recommendation score function f.

By differentiating the cost function in Eq. (3.7) with respect to f in order to obtain the optimal value, the MAP estimation $\hat{f}$ must satisfy $\lambda(\hat{f} - y) + \Delta\hat{f} = 0$. This can be solved either directly, or with the use of an off-the-shelf iterative method.

3.3 Heterogeneous Network Model

One of the main problem of treating a multi-domain heterogeneous network by a homogeneous network model is that *the similarity measures for either within-domain or cross-domain nodes are usually not comparable.* For example, the (visual) similarity of two images is not comparable to the

(semantic) similarity of two texts or the social similarity of two people. Moreover, it is questionable to treat the different kinds of cross-domain linkages with the same weight. For example, in the context of a recommender system, where images are being recommended for users, the weight of a preference link between a user (from the social domain) to the image domain should be weighted much higher than the weight of a link between the image and text domain.

In order to extend the regularization model from homogeneous networks to the heterogeneous case, we associate recommendation scores with all the domains, denoted with a superscript i for the i-th heterogeneous domain. Thus, in principle, it is possible to recommend not only target objects, but also other like-minded actor nodes, who have similar preferences to a given target actor in the social network. The recommendation score vector f then consists of relevance scores from different domains, i.e., $f^{(i)} : V^{(i)} \to \mathbb{R}$. Similarly, the context-bias function $y^{(i)} : V^{(i)} \to \mathbb{R}$ can be defined for the different domains. The MAP estimate of f is then given as follows:

$$\begin{aligned} f^* &= \arg\max \log p(f^{(1)}, \ldots, f^{(k)} | y^{(1)}, \ldots, y^{(k)}) \\ &= \arg\max \log p(y^{(1)}, \ldots, y^{(k)} | f^{(1)}, \ldots, f^{(k)}) p(f^{(1)}, \ldots, f^{(k)}) \\ &= \arg\max \sum_{i=1}^{k} \left(\log p(y^{(i)} | f^{(i)}) \right) + \log p(f^{(1)}, \ldots, f^{(k)}) \\ &= \arg\min \sum_{i=1}^{k} \left(\frac{\sigma}{2\mu^{(i)}} \| y^{(i)} - f^{(i)} \|^2 \right) + \tilde{\Omega}(f^{(1)}, \ldots, f^{(k)}) \end{aligned} \tag{3.8}$$

Here, $\tilde{\Omega}(f^{(1)}, \ldots, f^{(k)})$ is a joint smoothness functional across the domains. In heterogeneous networks, the smoothness functional should consider not only the smoothness of f at each domain, but also the linkages among the heterogeneous domains. In particular, we set the joint smoothness functional as follows:

$$\begin{aligned}\tilde{\Omega}(f^{(1)},\ldots,f^{(k)}) &= \frac{1}{2}\sum_{i=1}^{k}\sum_{v\in V^{(i)}}\|\nabla_v f\|^2 \\ &= \frac{1}{2}\sum_{i=1}^{k}\sum_{v\in V^{(i)}}\sum_{u\sim v,u\in V^{(i)}}\left(\frac{\partial f^{(i)}}{\partial e(u,v)}\bigg|_v\right)^2 \\ &+ \frac{1}{2}\sum_{i=1}^{k}\sum_{v\in V^{(i)}}\sum_{j\neq i}\sum_{u\sim v,u\in V^{(j)}}\left(\frac{\partial f^{(i)}}{\partial e(u,v)}\bigg|_v\right)^2 \\ &= \sum_{i=1}^{k}\left(\Omega(f^{(i)}) + \sum_{j\neq i}\Omega^H(f^{(i)},f^{(j)})\right) \end{aligned} \tag{3.9}$$

Here, $\Omega^H(f^{(i)}, f^{(j)})$ is the heterogeneous smoothness functional across domain i and domain j.

We then substitute Eq. (3.9) into Eq. (3.8) in order to obtain the MAP estimate of f.

$$\begin{aligned}\hat{f} = \arg\min \sum_{i\in\mathcal{D}}\left(\frac{\lambda^{(i)}}{2}\|y^{(i)} - f^{(i)}\|^2 + \Omega(f^{(i)})\right) \\ + \sum_{i\in\mathcal{D}}\sum_{j\sim i}\Omega^H(f^{(i)},f^{(j)})\end{aligned} \tag{3.10}$$

where $\lambda^{(i)} = \frac{\sigma}{\mu^{(i)}}$. So far, we have not discussed how the heterogeneous smoothness function Ω^H is defined. We will discuss it next.

3.3.1 Heterogeneous Smoothness Functional

As discussed earlier in Section 3.2, the smoothness enforcement on node values is used as an approach to achieve network structural propagation of the recommendation score values on the nodes. In this section, we will discuss how to extend this enforcement across domains, so that the knowledge of one domain can be transferred to another domain by enforcing the smoothness across domains, thus named *heterogeneous smoothness functional*. We note that the heterogeneous smoothness function is not the same as the homogeneous case. For a heterogeneous network, we expect that different content similarity measures are used in

different domains in order to construct the content-based linkages. For example, in the image domain we would adopt visual similarity, whereas in the text domain, we would use a term frequency – inverse document frequency (tf–idf) based feature similarity. As a result, *different domains may have different data distributions*, in terms of linkage weights and/or recommendation scores, and therefore it does not make sense to apply the same degree of smoothness enforcement across different domains.

To extend the smoothness function for heterogeneous networks, we put an emphasis on *cross-domain edges* in our smoothness model. For example, let us consider a cross-domain edge e between $u \in V^{(i)}$ and $v \in V^{(j)}$. We then define the cross-domain edge derivative of function $f^{(i)}$ along $(v, u) \in e^H$ at the node v as follows:

$$\left.\frac{\partial f^{(i)}}{\partial e^H}\right|_v = \sqrt{g_u(v)} f^{(j)}(u) - \sqrt{g_v(u)} f^{(i)}(v) \tag{3.11}$$

Here, $g(u)$ is the weighted degree function of node $u \in V^{(j)}$ based on the cross-domain weight $w(u, v)$, which can be either based on the links between one node in each domain (e.g., favor link between a user and an item), or based on the content similarity. In particular, $g_u(v) = \sqrt{\frac{\pi_{ij}(w(u,v))}{n_u}}$. Here, n_u is the normalization factor such that $n_u = \sum_{u \sim v, u \in V^{(k)}} \pi_{ik}(w(u, v))$. One critical issue of the cross-domain weights is that we might adapt different types of link or similarity measurements between different types of domains. For example, the rating links between users and images may be a number ranging from 1 to 5 while the cross-domain similarity between images and text may be a cosine similarity ranging from -1 and 1. Figure 3.2 illustrated such a distributional difference among the heterogeneous links. To handle such an issue we adapt a *weight normalization function* $\pi_{ij}(.)$ for capturing the variation in smoothness across different domains. In this proposal, we use two definitions for the weight normalization function. The first one is based on the z-normalization:

$$\pi_{ij}(w) = \ell \left(\frac{\sigma^{(i)}}{\sigma^{(ij)}} (w - \mu^{(ij)}) + \mu^{(i)} \right) \tag{3.12}$$

where $\mu^{(i)}$ and $\sigma^{(i)}$ are the mean and the standard deviation of the weights

in domain i, respectively. The $\ell(.)$ is a mapping function to make the weight ranging between 0 and 1. We use the logistic function as $\ell(.)$. The second choice of the weight normalization function is based on the percentile normalization:

$$\pi_{ij}(w) = \ell\left(q_i^{-1}\left(q_{ij}(w)\right)\right) \tag{3.13}$$

where q_i is the function that outputs the percentile of the given input based on the weight distribution of domain i and q_i^{-1} is the inverse function of q_i that returns the value which corresponds to the input percentile based on the weight distribution of domain i.

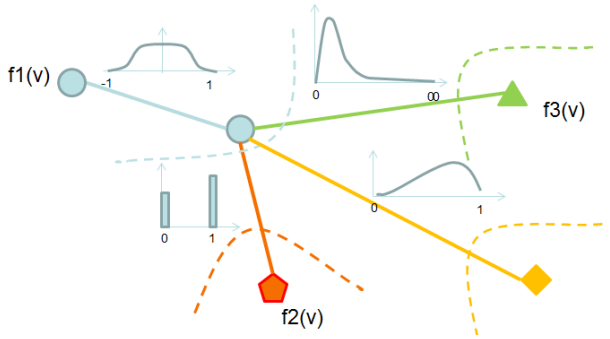


Figure 3.2: Variation of distribution among different types of heterogeneous links

With the cross-domain edge derivative it is not too difficult to rewrite the heterogeneous smoothness function with the *heterogeneous smoothness matrix* Φ as follows:

$$\Omega^H(f^{(i)}, f^{(j)}) = {f^{(i)}}^T \Phi^{(ij)} f^{(j)} \tag{3.14}$$

The *heterogeneous smoothness matrix* Φ is defined as follows:

$$(\Phi^{(ij)} f^{(j)})(v) = -\sum_{u \in V^{(j)}, u \sim v} \sqrt{g_u(v) g_v(u)} f^{(j)}(u) \tag{3.15}$$

Hence, we have the following:

$$\left.\frac{\partial \Omega^H(f^{(i)}, f^{(j)})}{\partial f^{(i)}}\right|_v = (\Phi^{(ij)} f^{(j)})(v)$$
$$= - \sum_{u \in V^{(j)}, u \sim v} \frac{\pi_{ij}(w(u,v))}{\sqrt{n_u n_v}} f^{(j)}(u) \tag{3.16}$$

3.4 Iterative Learning Algorithm

We note that the key to learning the recommendation score is to solve Eq. (3.10), which defines the posterior probability of the recommendation score. We note that Eq. (3.10) is expressed as a sum of similar terms across the different heterogeneous domains. Therefore, we first consider a subproblem of Eq. (3.10), in which we isolate the terms for a particular domain i. For a given domain i, we try to solve the following problem:

$$\min_{f^{(i)}} \frac{\lambda^{(i)}}{2} \|y^{(i)} - f^{(i)}\|^2 + \Omega(f^{(i)}) + \sum_{j \sim i} \Omega^H(f^{(i)}, f^{(j)}) \tag{3.17}$$

By differentiating the above cost function with respect to $f^{(i)}$ and setting it to 0 (for the optimal value point), the solution to Eq. (3.17) would satisfy the following:

$$\lambda^{(i)}(f^{(i)} - y^{(i)}) + \Delta f^{(i)} + \sum_{j \sim i} \Phi^{(ij)} f^{(j)} = 0 \tag{3.18}$$

As this equation plays an important role of obtaining the recommendation scores, we call it an *update function*. We abbreviate one of the terms in Eq. (3.18) by $z^{(i)}$.

$$z^{(i)} = - \sum_{j \sim i} \Phi^{(ij)} f^{(j)} \tag{3.19}$$

We further note that the Laplace operator $\Delta f^{(i)}$ on the weighted adjacency matrix $S^{(i)}$ for the homogeneous sub-network of the i-th domain can be written as $\Delta f^{(i)} = (I - \Phi^{(i)}) f^{(i)}$. Therefore when the heterogeneous

smoothness matrix $\Phi^{(ij)}$ is given, the solution to Eq. (3.17) can be explicitly written as follows:

$$\begin{aligned}\hat{f}^{(i)} &= (I - \alpha\Phi^{(i)})^{-1}\left((1-\alpha)y^{(i)} - \alpha\sum_{j\sim i}\Phi^{(ij)}f^{(j)}\right) \\ &= (I - \alpha\Phi^{(i)})^{-1}\left((1-\alpha)y^{(i)} + \alpha z^{(i)}\right) \end{aligned} \tag{3.20}$$

where $\alpha = \frac{1}{1+\lambda^{(i)}}$. Now, we present a two-step algorithm for solving the heterogeneous network problem (3.10) in Algorithm 1.

The algorithm assumes that the input to the algorithm contains a recommendation based query $Q = \{q_j^{(i)} | i = 1, \ldots, N, j = 0, \ldots, |q^{(i)}|\}$. This query specifies the target user (or target object), and corresponding contextual components for the recommendation process, such as an image or a keyword. The input of the algorithm also includes the graph G with its structure and contents, as well as two parameter sets λ's and γ's. An integral component of using content in the recommendation process is to be able to compute similarities between nodes. These similarity values are needed for construction of the *intra-domain* similarity matrix S, as well as the initial recommendation target vector y. The similarity measure $sim(u, v)$ between two nodes u and v depends upon the kind of domain relevant to the nodes u and v. For example, in in the text domain, a tf–idf cosine similarity is used, whereas in the image domain a visual similarity is used. We will define these similarity measures more explicitly in the experimental section (Section 3.5).

The weighted adjacency matrix S^i is defined in terms of the similarity between nodes $u, \bar{u}, v, \bar{v} \in V^{(i)}$ of the same kind:

$$S^i(u, v) = \frac{sim(u, v)}{\sqrt{\sum_{\bar{u}\sim u} sim(\bar{u}, u) \sum_{\bar{v}\sim v} sim(\bar{v}, v)}} \tag{3.21}$$

The recommendation target vector $y^{(i)}$ is defined as the average similarity between a node v and all components of the recommendation query Q, which lie in domain i. Therefore, for domain i we have:

$$q^{(i)}(v) = \frac{1}{Z_q} \sum_{u \in C^{(i)}} sim(u, v), \quad \forall v \in V^{(i)} \tag{3.22}$$

and

$$y^{(i)} = \xi \frac{1}{N} \mathbf{1} + (1 - \xi) q^{(i)} \tag{3.23}$$

Here, $C^{(i)}$ is the contextual components related to i-th domain and Z_q is used to normalize the score vector $q^{(i)}$. We note that $y^{(i)}$ may not be defined for some of the nodes in the network. For example, for a recommendation query involving only a user, the value of $y^{(i)}$ may be undefined for nodes in the image and text domain. Furthermore, the value of $y^{(i)}$ may be quite inaccurate for nodes in the social domain, which are not directly related to the relevant user. The regularization process discussed in this proposal essentially serves the goal of handling the noisiness and inaccuracies in our initial estimate $y^{(i)}$ with the use of a regularization process.

We also initialize the heterogeneous smoothness matrices to identity matrices with the graph structure. After initialization, we start the algorithm by iteratively processing each homogeneous network to obtain the recommendation scores of nodes in the network based on the heterogeneous smoothness matrices which take the heterogeneous linkage and recommendation scores of other domains into account. The second step of each iteration is to update the heterogenous smoothness matrices with the newly obtained recommendation score vector of the current domain. The algorithm is terminated when the recommendation scores converge, or the maximum number of iteration have been achieved.

3.4.1 Convergence Analysis

Next, we will show the convergence of the proposed algorithm, which uses iterative re-computation of the parameters across different homogeneous subnetworks. For simplicity, we use the case where the heterogeneous network G has only two homogeneous sub-networks, $G^{(i)}$ and $G^{(j)}$. The steps of the proof can be easily generalized to the case where there are more than two sub-networks. We also assume that $G^{(i)}$ and $G^{(j)}$ are each a

Algorithm 1 Heterogeneous Recommendations with Graph Regularization

input query Q, heterogeneous graph G, parameters $\lambda^{(i)}$
1. Initialization (for all domains):
   weighted adjacency matrix S
   heterogeneous smoothness matrix Φ
   contextual-bias score y
2. Set $t = 1$
   repeat
     for each homogeneous sub-network i **do**
3. STEP 1: Obtain $z^{(i)}$ by Eq. (3.19)
4. STEP 2: Solve i-th subproblem (3.17) using Eq. (3.20)
     end for
5. $t = t + 1$
   until Convergence on $f^{(i)}$ or t achieves maximum iteration number.
output $f^{(i)}$ for each domain

strongly connected graph, or equivalently, $\Phi^{(i)}$ and $\Phi^{(j)}$ are irreducible, hence their invertible structural matrices $(I - \alpha\Phi^{(i)})^{-1}$ and $(I - \alpha\Phi^{(j)})^{-1}$ exist and are unique. This assumption can be easily hold by further decomposing a graph that is not strongly connected into several subgraphs that are each strongly connected.

Now consider the recommendation scores on $G^{(i)}$ with the within-domain matrix $\Phi^{(i)}$ and the cross-domain matrix $\Phi^{(ij)}$. Then the update function for $f^{(i)}$ is

$$f^{(i)} = (I - \alpha\Phi^{(i)})^{-1} \left((1 - \alpha)y^{(i)} - \alpha\Phi^{(ij)} f^{(j)}\right) \tag{3.24}$$

and similarly for $f^{(j)}$ we have

$$f^{(j)} = (I - \alpha\Phi^{(j)})^{-1} \left((1 - \alpha)y^{(j)} - \alpha\Phi^{(ji)} f^{(i)}\right) \tag{3.25}$$

By substituting Eq. (3.25) into Eq. (3.24) we obtain

$$f^{(i)} = (I - \alpha\Phi^{(i)})^{-1} \left(Af^{(i)} + (1 - \alpha)y^{(i)}\right) + By^{(j)} \tag{3.26}$$

with

$$\begin{aligned} A &= \alpha^2\Phi^{(ij)}(I - \alpha\Phi^{(j)})^{-1}\Phi^{(ji)} \\ B &= -\alpha(1 - \alpha)(I - \alpha\Phi^{(i)})^{-1}\Phi^{(ij)}(I - \alpha\Phi^{(j)})^{-1} \end{aligned} \tag{3.27}$$

From the power iteration in Eq. (3.26) we can see that the algorithm will converge to a unique solution for $G^{(i)}$ due to the irreducibility of $\Phi^{(i)}$. This follows the assumption that $G^{(i)}$ is strongly connected.

To see this, we let $y^{(i)} = \xi\frac{1}{N}\mathbf{1} + (1-\xi)q_u$, with q_u being the user-specific vector (i.e., the entry corresponding to the user for recommendation is 1 and otherwise 0, if it is the user domain). Here, ξ is the balancing parameter which is set to 0.1 for the user domain and set to 1 for the other domains. Note that the setting of $y^{(i)} = \xi\frac{1}{N}\mathbf{1} + (1-\xi)q_u$ is only for the ease of illustration. In reality, $y^{(i)}$ can be any positive stochastic vector, i.e., all the elements in $y^{(i)}$ are positive and sum to 1. We then reorganize Eq. (3.26) as

$$\begin{aligned} f^{(i)} &= (I - \alpha\Phi^{(i)})^{-1}\left(Af^{(i)} + (1-\alpha)y^{(i)}\right) + By^{(j)} \\ &= (I - \alpha\Phi^{(i)})^{-1}\left(A + \frac{\xi(1-\alpha)}{N}\mathbf{1}\mathbf{1}^{\mathsf{T}}\right)f^{(i)} + b \end{aligned} \tag{3.28}$$

with $b = By^{(j)} + (1-\xi)(1-\alpha)(I - \alpha\Phi^{(i)})^{-1}q_u$.

We note that the matrix $(I - \alpha\Phi^{(i)})^{-1}\left(A + \frac{\xi(1-\alpha)}{N}\mathbf{1}\mathbf{1}^{\mathsf{T}}\right)$ is irreducible because $\Phi^{(i)}$ and $\mathbf{1}\mathbf{1}^{\mathsf{T}}$ are irreducible and A is non-negative, therefore $f^{(i)}$ will converge to a unique solution. Similarly, one can show that $f^{(j)}$ converges to a unique solution as well. Hence the convergence of the algorithm is guaranteed.

3.5 Experimental Results

In this section, we present the recommendation effectiveness results with the use of this approach. One main advantage of our approach is that it uses a careful combination of network structure, linkages and content-similarity for the recommendation process. This provides it an advantage over other competing methods which do not use the rich information available in a multimedia information network.

3.5.1 Preliminary

Data Sets

We evaluate our proposed method with two data sets with different characteristics. The first data set is *MovieLens100K* which is a public and very popular data set in the recommender system community. This data set is of small scale and uses a 5-scale rating system. However, *MovieLens100K* only provides limit side information on the users and items (i.e., movies).

On the other hand, to fully evaluate our proposed method that emphasizes on the utilization of the content information of items and social information of users, we collect a *Flickrgroup* data set from Flickr[1] that provides great side information on the items and users, such as descriptions, tags, user comments, user groups and so on. Also, the rating system of *Flickrgroup* data set is totally different from the 5-scale system of *MovieLens100K* data set. The rating system of *Flickrgroup* data set is a one-class system that only provides the positive ratings, which is also known as the *one-class collaborative filtering* (OCCF) problem [11, 12]. OCCF is very challenging and there is only a few literature comparing to that for the 5-scale rating system.

Effectiveness Measures

As stated in Section 3.1, our focus on the recommendation task is to provide a top-N list of items that the users would like. This is indeed a ranking problem but not a rating prediction problem. Therefore, we evaluate the methods with the ranking measures. For the *MovieLens100K* system, we are able to adopt the popular normalized Discounted Cumulative Gain (nDCG) measure as it is 5-scale rated so we have both positive (higher ones) and negative (lower ones) ratings available and can ignore the missing (0) ratings. However, for the *Flickrgroup* data set where only positive ratings are provided, it is very problematic to treat missing ratings as negatives when doing the evaluation. Therefore, we assess the performance with the percentile scores (PS) of the true positive items. The percentile score of an item is its ranking in the returned list in terms of percentage. Specifically, the percentile score of an

[1] http://www.flickr.com

item i in the returned list L is

$$PS(i) = 100 \times \frac{rank(i)}{|L|} \tag{3.29}$$

And for each method we report the mean average percentile scores (mAPS) as follows:

$$mAPS = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|TP(u)|} \sum_{i \in TP(u)} PS(i) \tag{3.30}$$

where $\mathcal{U}$ is the set of testing users and $TP(u)$ is the true positive favored items for user u.

3.5.2 Compared Algorithms

We compared our algorithm against three popular recommendation models, including PMF[10], WRMF [11, 12], SoRec [26], WRSoRec (a variation of SoRec), RWR [21] and GWNMTF [37]. Table 3.1 summaries the information used and the computational complexity of these methods as well as our proposed one.

- *Probabilistic Matrix Factorization* (PMF) [10] is a probabilistic latent factor-based approach proposed for collaborative filtering. PMF aims at fitting the user-item rating matrix using low-rank approximations, that is, the ℓ-dimensional user and item factor matrices (ℓ is small). The method, however, ignores the social activities and content information and simply assumes the users and items are independent and identically distributed. PMF has been considered a classical collaborative filtering method and does have fairly good performances with several recommendation data sets. However, it could not handle one-class collaborative filtering (OCCF) problem since the use of indicator matrix I makes it either ignore all missing ratings or treat them as negative.

- *Weighted Regularized Matrix Factorization* (WRMF) is a matrix factorization algorithm proposed by by [11] and [12] to handle the one-class collaborative filtering (OCCF) problem, including the

implicit feedback such as clickthroughs or the bookmark information from users. The favorite images of the users in the *Flickrgroup* data set also belongs to the one-class data set because it is binary. WRMF, as a variation from the PMF method, assumes that *missing ratings are most likely to be negative ratings* and therefore it sets the missing ratings to be zero and adopts a weight matrix to put small weights on those missing ratings and higher weights on true positive ratings. This trick along with AMAN (all missing as negative) successfully resolves the problem of PMF which poorly handles OCCF by AMAU (all missing as unknown). In this experiment, we set the weight matrix as follows:

$$w_{ui} = \begin{cases} 1+\alpha, & \text{if } r_{ui} = 1 \\ 1, & \text{otherwise} \end{cases} \tag{3.31}$$

and $\alpha \in \{0.1, 0.3, 0.5, 1, 2\}$ and the dimension of the latent factors is set to {5,10,20,30} while the regularization parameters λ_U and λ_V are set equally to {0.01, 0.1, 1}.

- *SoRec* [26] is an extended version of PMF which fuses the user-item rating matrix with the user's social network (C). The tuition behind the approach that people in a user's social network affect the user's personal behaviors more than those not in the user's network do. SoRec has experimentally shown an average 9.98% improvement over PMF on the Epinions data set while maintaining the same computational complexity. Specifically, SoRec has an objective function as Eq. (3.32):

$$\begin{aligned} \min_{U,V,Z} \frac{1}{2}\|I^R \circ (R - g(U^\mathsf{T}V))\|_F^2 + \frac{\lambda_C}{2}\|I^C \circ (C - g(U^\mathsf{T}Z))\|_F^2 \\ + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2 \end{aligned} \tag{3.32}$$

  The dimension of the all three latent factors (for user, item and social) is set to {5,10,20,30}. The regularization parameters λ_U, λ_V and λ_Z are set equally to {0.01, 0.1, 1} while λ_C is set to 5 times of λ_U.

- *WRSoRec*: As we noted in WRMF method, the PMF method is not designed for handling the OCCF problem, we employ the trick from WRMF to the SoRec such that the missing ratings are not ignored but

properly weighted. We call this variation method *WRSoRec*, which has an objective function as Eq. (3.33).

$$\min_{U,V,Z} \frac{1}{2}\|W \circ (P - g(U^\mathsf{T}V))\|_F^2 + \frac{\lambda_C}{2}\|I^C \circ (C - g(U^\mathsf{T}Z))\|_F^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2 \tag{3.33}$$

where P is of the same dimension as R and is a binary matrix whose entries are 1 if the corresponding entries in R is positive and are 0 otherwise. In the experiment with *Flickrgroup*, we set W the same as that in Eq. (3.31). The dimension of the all three latent factors (for user, item and social) is set to {5,10,20,30}. The regularization parameters λ_U, λ_V and λ_Z are set equally to {0.01, 0.1, 1} while λ_C is set to 5 times of λ_U.

- *Random Walk with Restart* (RWR) is the adaptation of a recent approach which uses social network analysis for the recommendation process [21]. However, this algorithm does not use any content information, and uses a homogeneous network model for the recommendation process. The algorithm constructs a model in which three types of nodes are constructed for image nodes, actor nodes and tag nodes. The only linkages in this network correspond to friendship, tagging, upload or likeability relationships. For the case of user-centered recommendations, the RWR algorithm uses a restart vector to indicate the user node of interest.

- *Graph regularized weighted non-negative matrix tri-factorization (GWNMTF)* [37] is one of the state-of-the-art methods that adds regularization constraints on the user and item graph so that the profiles of similar users/items are closer in the latent factor space. Similar to PMF, GWNMTF ignores the missing ratings and could not handle the OCCF problem.

Table 3.1: Comparison of the proposed model and the others

Model	Information Used			Time Complexity		Handle OCCF
	ratings	social	content	training	testing	
PMF	Y	-	-	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell n)$	-
WRMF	Y	-	-	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	Y
SoRec	Y	Y	-	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell n)$	-
WRSoRec	Y	Y	-	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	Y
CTR-SMF	Y	Y	LDA	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	-
RWR	Y	Y	link-based	–	$\mathcal{O}(k^2 n^2)$	Y
GWNMTF	Y	Y	link-&sim-based	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	-
HNM	Y	Y	link-&sim-based	$\mathcal{O}(kn^3)$	$\mathcal{O}(n^2)$	Y

Table 3.2: Statistics of *MovieLens100K* data set

ratings	1	2	3	4	5	Total
# ratings	6,110	11,370	27,145	34,174	21,201	100,000
# users	723	894	936	942	928	943
# item	1,363	1,325	1,470	1,384	1,172	1,682

3.5.3 *MovieLens100K* Data Set

Background and Features

The *MovieLens100K* data set is prepared by GroupLens[2] and is publicly available for experiments on recommendation tasks. It consists of 100,000 ratings on 1,682 movies from 943 users. The ratings are of 1-5 scales indicating the strength of the likeness from the users to movies. A rating of 5 indicates the strongest likeness while 1 indicates the weakest likeness or the strongest dislikeness. Unknown or missing ratings are designated by 0. In addition to the ratings, it also provides simple demographic information about the users and the genre information about the movies. The demographic information includes the age, gender and occupation of the users while genre information includes 19 genres of the movies. The ratings are collected during a 7-month period in the late 1990s and has been cleaned up such that only users with more than 20 ratings and with complete demographic information are kept. Table 3.2 gives some statistics about the data set.

[2]http://www.grouplens.org/node/73

Table 3.3: Categories of the side information for *MovieLens100K* data set

age group	1:1-17; 2:18-24; 3:25-34; 4:35-44; 5:45-49; 6:50+
gender	0:female; 1:male
user occupation	1:administrator; 2:artist; 3:doctor; 4:educator; 5:engineer; 6:entertainment; 7:executive; 8:healthcare; 9:homemaker; 10:lawyer; 11:librarian; 12:marketing; 13:none; 14:other; 15:programmer; 16:retired; 17:salesman; 18:scientist; 19:student; 20:technician; 21:writer
movie genre	1:unknown; 2:action; 3:adventure; 4:animation; 5:children's; 6:comedy; 7:crime; 8:documentary; 9:drama; 10:fantasy; 11:film-noir; 12:horror; 13:musical; 14:mystery; 15:romance; 16:sci-fi; 17:thriller; 18:war; 19:western

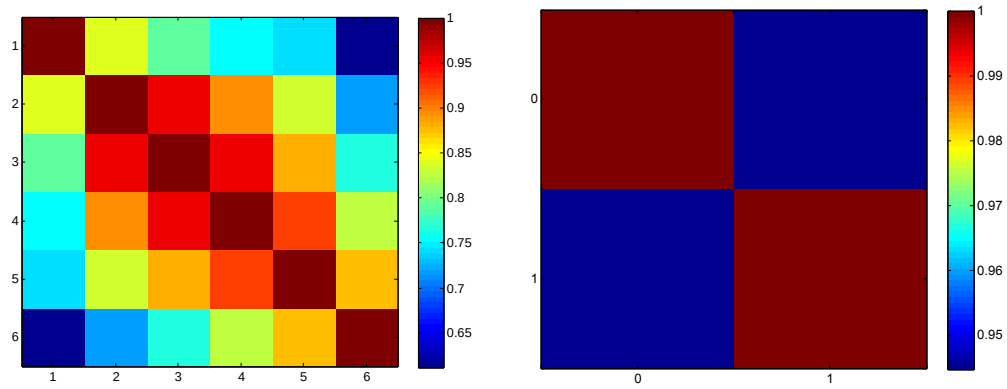
The side information for the users consists of the users' age, gender and occupation. We further partition the age into six age groups for the users: {1-17, 18-24, 25-34, 35-44, 45-49, 50+}. The categories of each side information are listed in Table 3.3.

To get some preliminary idea on how much the side information can help, we report the average rating similarities on each pair of groups of users from three different aspects as depicted in Figure 3.3.

To construct the heterogeneous network for the user and movie domain, we adopt the demographic similarity and the genre similarity. In particular, the weight of the links between two users depends on whether the two users fall in the same category of age group, gender and occupation. Each of the matches will increase the weight by $\frac{1}{3}$, starting from 0. So the maximum of the weight will be 1 and minimum will be 0. For the weight of the links between two movies depends on the genres of the two movies. Each movie is represented by a 19-dimensional binary vector indicating the genres that the movie belongs to. Note that a movie may belong to multiple genres. The weight of the links between two movies is then the cosine similarity of the two binary vectors representing the two movies.

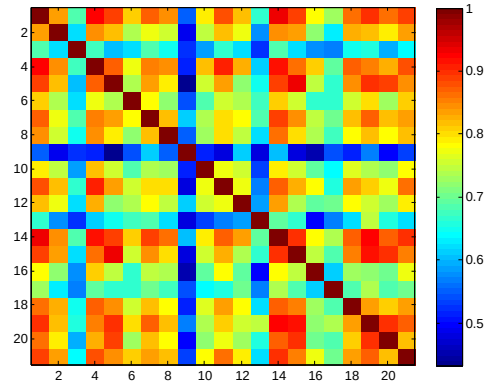
Experiment Setup

For the experiments on *Movielens100K* data set, we use an 80-20 training/testing rating split on the 100 K movie ratings of 943 users for 1682 movies. The weight of the user-user relationship is based on a

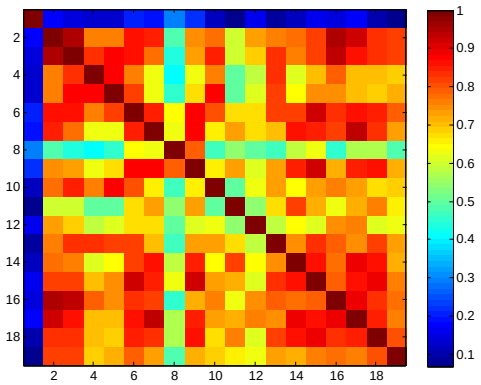


(a) User age group

(b) User gender



(c) User occupation



(d) Movie genre

Figure 3.3: Categorical similarities from different aspects

30-dimensional binary vector from the users' demographical information, namely age, gender and occupation. In particular, 6 out of 29 dimensions are used to indicate the age group that the user belongs to: {1-17, 18-24, 25-34, 35-44, 45-49, 50+}. And 2 out of 29 dimensions are to indicate the gender of the user. The remaining 21 dimensions are used to indicate the occupation of the user. If two users are of the same age group, the weight will increase by 1/3; and if two users are of the same gender, the weight will increase by another 1/3; and if two users are of the same occupation, the weight will increase by another 1/3. The minimum of the weight between two users is 0 when two users are not in the same age group and not of the same gender nor the occupation. The maximum of the weight between two users is 1 when two users are of the same age range, gender and occupation. The weight of the item-item relationship is based on the movie genres the item belongs to. There are in total 19 movie genres and one movie at least

Table 3.4: Comparison of MAE and nDCG on *MovieLens100K* data set

	MAE	nDCG	nDCG1	nDCG10	nDCG20	nDCG100
PMF	0.7577	0.7318	0.8605	0.7596	0.742	0.7224
SoRec	0.7501	0.747	0.8569	0.7783	0.7732	0.7497
RWR	0.8213	0.8304	0.8482	0.8425	0.8392	0.8439
GWNMTF	**0.7179**	0.8511	0.8659	0.8778	0.8715	0.8626
HNM	0.7925	**0.8719**	**0.9124**	**0.883**	**0.8825**	**0.8835**

belongs to one of these 19 genres. Each item is represented by a 19-dimensional binary vector where each entry of the vector indicates whether the item belongs to the genre. The weight between two items is then the cosine similarity of the two vectors of the items. Note that the minimum weight of the items is also 0. For the weight of user-item relationship, we simply use the normalized ratings where the ratings are scaled by 1/5.

Recommendation Effectiveness Results

We compare the proposed *HNM* method with four competing methods: PMF [10], SoRec [26], RWR [21] and GWNMTF [37]. The introduction of each method can be found in Section 3.5.2. We summarize the effectiveness performance in Table 3.4, where we observed that HNM does not perform well in terms of MAE metric, which is used to measure the performance of rating prediction. It is indeed expected as the HNM is designed to obtain a list of top-ranked items that the users would be interested in, therefore the relative scores among items are much more important than the absolute rating value of each item. On the other hand, HNM does outperform the competing methods in terms of nDCG metric, except for nDCG1. This indicates that HNM is a better choice for making top-K recommendations to the users than the other methods.

Efficiency Analysis

As summarized in Table 3.1, one of the main issue of the propose HNM method is its inefficiency. It requires high computational complexities for both training ($\mathcal{O}(kn^3)$) and testing ($\mathcal{O}(n^2)$). What we have observed in

the experiment on *MovieLens100K* data set matches this high complexity requirement for the HNM method, as listed in Table 3.5. The training time for the HNM method is about 15 times of that for the PMF method while the testing time is about 6,700 of that for the latent factor based methods. We will discuss more on how to reduce the computational complexity with HNM method in Chapter 4.

Table 3.5: Comparison of training and testing time used on *MovieLens100K* data set (in seconds)

	Training	Testing
PMF [10]	34.19	0.01
SoRec [26]	70.67	0.01
RWR [21]	-	491.37
GWNMTF [37]	177.85	0.03
HNM	528.10	67.88

3.5.4 *Flickrgroup* Data Set

Background and Features

The *Flickrgroup* data set is a social media data set containing information from three domains, namely text, image, and social domain. The key elements of the social domain on Flickr are the users and the groups. The users are those who upload, comment, tag, favor the images on Flickr. The groups, on the other hand, are communities with people who have the same interests toward a target subject. The group members typically favor photos which are closely related to the target subject. Therefore, we use this membership relation for user similarities where, in particular, users in the same group are similar to each other. In the *Flickrgroup* data set, there are about 34,640 users from 139 groups, whose members favor more than 2.2 million images in total. The average number of users that one image is favored by is 1.582. There are also more than 1,470,000 unique tags associated with these images. To pick the tags with the most discriminative power, we employ stop-word removal and stemming, and then select the top 5,000 most frequent ones as the codebook tags.

For the image representation, we adopt the Hierarchical Gaussianization (HG) [43], where each image is represented by a Gaussian mixture model based super-vector. Specifically, all the images are first resized to maximum 240 $\times$ 240 and segmented into squared patches with three different sizes: 16, 25 and 31, by a 6-pixel step size. The 128-d Histogram of Oriented Gradients (HOG) feature is then extracted from each patch and followed by a PCA dimension reduction to 80-d. To obtain the feature characteristics of the image collection, we randomly select the patches from the whole image collection and learn a Gaussian mixture model (GMM) with 512 components, named the Universal Background Model (UBM). With the trained UBM, we then can obtain the statistics of the patches of an image by adapting the distribution of the extracted HOG features from these image patches to the UBM. The final dimension of the image feature is 42,496.

Homogeneous links. For three types of domains we adopt different homogeneous linkage definitions. For the user domain, the linkage similarity is defined based on the membership of the users. In particular, the similarity of the edge between user U_A and U_B is as follows:

$$sim(U_A, U_B) \propto \left| Grp(U_A) \bigcap Grp(U_B) \right| \tag{3.34}$$

where $Grp(U)$ represents the groups that user U belongs to. For the text domain the linkage similarity is defined based on the co-occurring words of the texts. Thus, the similarity between the text nodes T_A and T_B is:

$$sim(T_A, T_B) = \frac{\langle \delta(T_A), \delta(T_B) \rangle}{\|\delta(T_A)\| \|\delta(T_B)\|} \tag{3.35}$$

where $\delta(T)$ is the bag-of-word representation of T based on the 5000-d text codebook. For the image domain, the similarity is defined based on the visual content similarity of the images. Specifically, the similarity between image I_A and I_B is

$$sim(I_A, I_B) = h\left(\frac{\langle \rho(I_A), \rho(I_B) \rangle}{\|\rho(I_A)\| \|\rho(I_B)\|} \right) \tag{3.36}$$

where $\rho(I)$ is the 42,496-d GMM-based super-vector of image I and $h(x) = \frac{1}{2}(x+1)$ maps the score from [-1, 1] to [0, 1].

It is worth noting that the similarity scores defined above are all

non-negative which makes the within-domain matrices stochastic matrices hence the convergence of our algorithm is guaranteed. Also, to maximize effectiveness, all edges are assumed to be bidirectional.

Experiment Setup

In order to predict the effectiveness of the method, we will use the *"like"* feedback provided by users as a binary rating which is predicted by the algorithm. About 10% of the *"like"* ratings were removed from the data for testing purposes, and 90% of the *"like"* ratings were retained within the network for training purposes (as links in the network). We denote E^{like} as the set of the *"like"* edges which are divided into two non-overlapping training and testing sets, denoted as E^{like}_{train} and E^{like}_{test}, respectively. Of course, a variety of other linkage and contextual information is also available for training. One challenge with such a prediction algorithm is that the ratings are extremely *sparse*, and therefore the absence of a *"like"* link does not necessarily imply that the user does not like the image. Therefore, the evaluation needs to focus on the specific rankings of images for which user rating is available, rather than those for which it is absent.

We note that we have two kinds of social recommendations: *user-centered*, in which images are recommended for a specific user X, and *image-centered*, in which users are recommended for a specific image Y. In some of the cases, additional context may also be available, though the cores of these methods either recommend objects or they recommend users. In each of these cases, the evaluation of the algorithm is performed as follows:

- For the case of user-centered recommendations, we rank *all* the images by order of the algorithm recommendation score for a relevant user X. Among these ranked images, we determine those for which the user X has exhibited a *"like"* preference in the *test data*. Specifically, we evaluate with the images which linked to the user X by the *"like"* link set E^{link}_{test}. We report the *mean average percentile score* (mAPS) of the ranked images which are indeed preferred by the user X, based on the preferences expressed in the *test data*. We note that the lower the APS, the better the algorithm.

- For the case of image-centered recommendations, we rank *all* the users

in order of recommendation score for a specific image Y. The same approach as above is used in order to compute the average percentile score.

The results were averaged over 1000 target users (or target images for image-centered queries) which were also selected from the Flickr network. We also performed some *context-sensitive recommendations*, in which image recommendations were provided for a user along with one of their uploaded images (or tagged keywords) as a context. As before, we used 1000 queries in this case, and averaged the percentile score. For the parameter setting of our proposed method, we set the smoothness and target closeness tradeoff parameter $\lambda^{(i)} = \frac{1}{9}$ for all i, and the regularized parameter in the target vector $\xi = 0.1$.

Recommendation Effectiveness Results

In this experiment, we use a subset of Flickgroup by selecting the images which are favored by more than 10 users. This results in 9,699 images and 20,298 users with around 158,000 favor links. 1000 query users are randomly selected out of 20,298 users. We use a 50/50 train/test split on the image collection that contains 9,699 images. The scenario is that we remove the "like" links between the 1000 query users and the testing images and our goal is to recover these links. Note that since only positive preference links are available, *Flickrgroup* is indeed a OCCF data set. Therefore, for a fair comparison we only compare our model with the competing algorithms that can handle OCCF data set, namely WRMF, WRSoRec and RWR. As shown in Table 3.6, the proposed HNM model improves over the other models (WRMF, WRSoRec and RWR) by 27.2%, 18.0% and 22.7%, respectively. In addition to an average of the percentile scores over the testing images, for each algorithm we also plot a P-PS curve such that a point on the curve gives the percentage of favored images from the testing data (Y-axis) falling within a specific percentile score in the rank (X-axis), as shown in Figure 3.4. A higher curve implies that a larger percentage of the relevant images lie within a specific percentile score, and this is desirable.

Table 3.6: Performance comparison on *Flickrgroup* data set with different methods

	mAPS	Prec@1	Prec@5	Prec@10	Prec@20
WRMF	38.42	0.0160	0.0124	0.0097	0.0077
WRSoRec	34.12	0.0280	0.0182	0.0141	0.0104
RWR	36.18	0.0130	0.0116	0.0083	0.0066
HNM	**27.98**	**0.0440**	**0.0220**	**0.0157**	**0.0118**

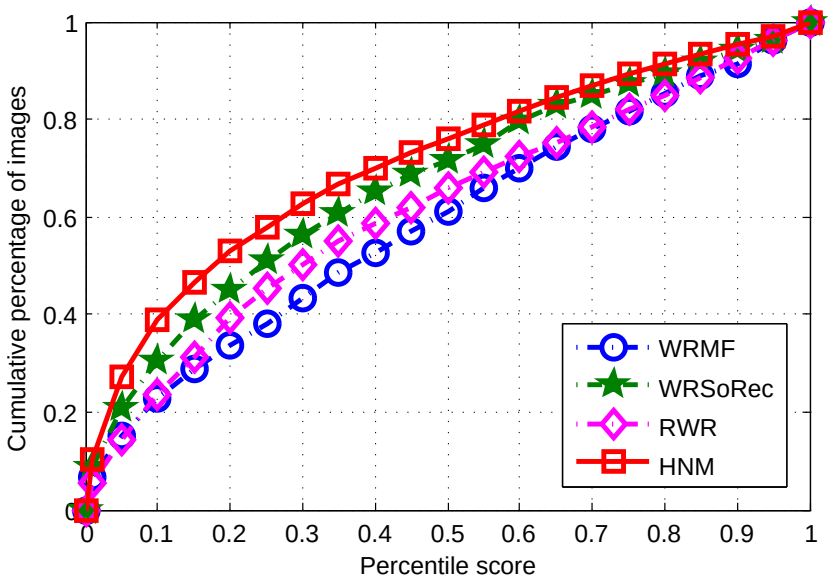


Figure 3.4: P-PS curve of image recommendation for specific users

Efficiency Analysis

Now we compare the four approaches with respect to the efficiency of training and testing. The testing time reported is the total time used for computing the recommendation scores for 1000 users. All experiments were performed on a machine equipped with an Intel Core i5 2.3 GHz CPU (Dual Core) and 8 GB memory.

As shown in Table 3.7, the latent factor based approaches are very efficient which not only take less than 500 seconds to train the latent user and item matrices, but also require 0.3 seconds to compute the recommendation scores for all the users (as it is simply a matrix multiplication). On the other hand, the RWR method does not require a training phase due to the adaptation of power iteration method but it takes around 1.4 seconds for 100 iteration to obtain the recommendation score for each user. Finally, although our

proposed method HNM outperforms all the other three algorithms in terms of mAPS, it does take much longer time for both training and testing comparing with the latent factor models. In all, this experimental observation matches our complexity analysis on the methods earlier and there does exist quite a lot space for improvement for our proposed method. We will discuss this more in Chapter 4.

Table 3.7: Comparison of training and testing time used for four algorithms (in seconds)

	Training	Testing
WRMF	109	0.3
WRSoRec	462	0.3
RWR	-	1384.7
HNM	1536	261.2

Impact of Parameter λ

In our proposed HNM model, parameter $\lambda^{(i)}$ balances the smoothness of the recommendation scores $f^{(i)}$ across the networks and the closeness between the recommendation scores $f^{(i)}$ and the target query vector $y^{(i)}$. At the extreme case, if $\lambda^{(i)}$ is set close to 0, we will emphasize the smoothness over the closeness to the target vector. As a result, the obtained recommendation scores would just depend on the eigestructure of the graph and not have personalized results. On the other hand, if $\lambda^{(i)}$ is set to a very large number, the recommendation scores will be the target scores, which is trivial. In other cases, we seek a balance to fuse the closeness requirement for personalization and the smoothness for social and content regularization.

Table 3.8 summarizes mAPS comparison for different settings in terms of the amount of training data and the tradeoff parameter λ ,while Figures 3.5, 3.6 and 3.7 illustrates the impact of λ with the P-PS curves when using 20%, 50% and 80% of the users for training. It is clearly that with more users for training the performance would be better. Also, one can observe that for the Flickgroup data set, our HNM approach achieves the best performance when λ is set to $\frac{1}{9}$, while smaller values (e.g., $\frac{1}{99}$) or larger values (e.g., 9) would largely decrease the performance. This observation is quite consist over different percentage settings of the number of training users.

Table 3.8: mAPS comparison on *Flickrgroup* data set with different experimental settings

	$\lambda = 9$	$\lambda = 1$	$\lambda = \frac{1}{9}$	$\lambda = \frac{1}{99}$
20% training users	49.68	45.95	43.46	47.42
50% training users	40.71	29.57	27.98	38.55
80% training users	37.51	25.67	24.86	32.05

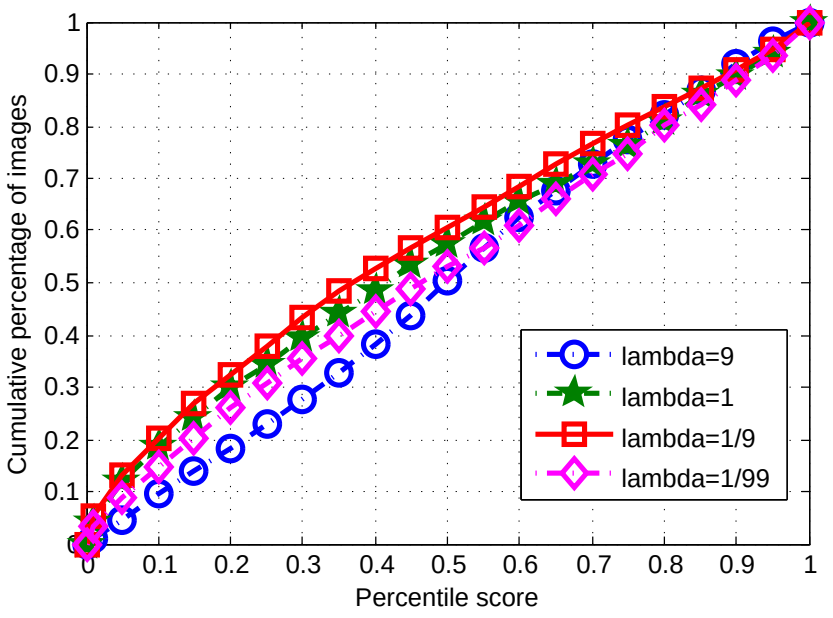


Figure 3.5: Impact of parameter λ when using 20% users for training

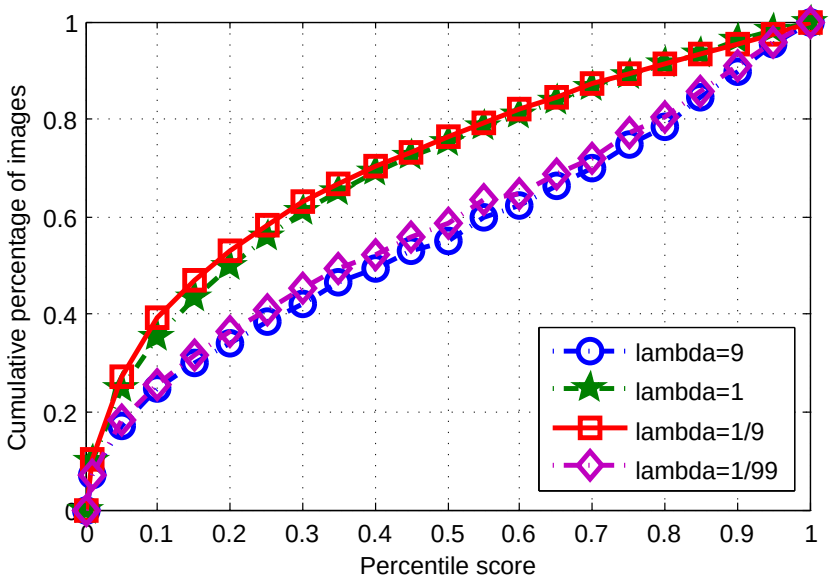


Figure 3.6: Impact of parameter λ when using 50% users for training

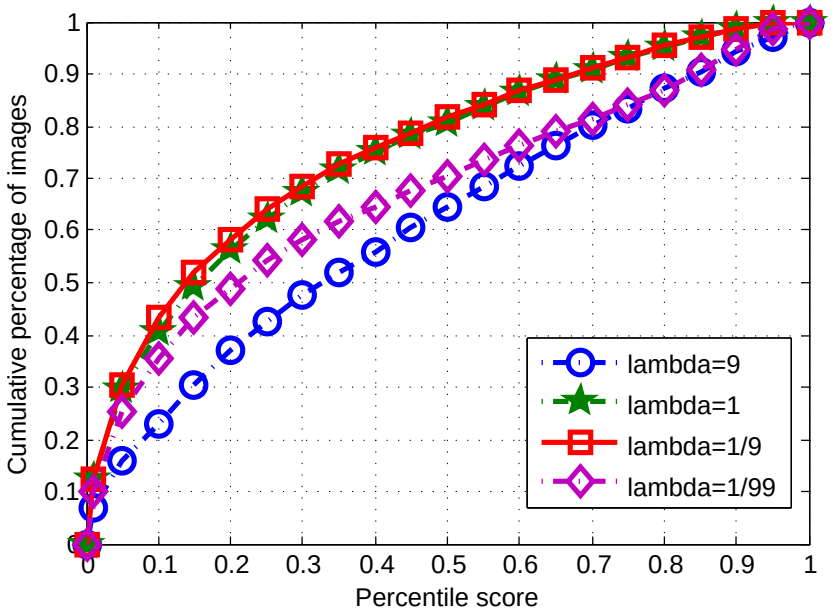


Figure 3.7: Impact of parameter λ when using 80% users for training

CHAPTER 4

HYBRID RECOMMENDATION MODEL

In the literatures, latent factor models have shown their power in effectively estimating overall structure with low-rank approximations [2, 10, 26]. The idea behind such models is that the relations (e.g., rating, friendships, memberships, ...) are usually determined by a small number of unobserved factors. Recent advances in efficient training of the latent factor models have promoted the approaches in many applications, including rating prediction and link structure discovery. In these applications, the latent factors are used to capture the intrinsic structure of the target relations among entities or nodes. In particular, SVD++ [2] not only achieves the state-of-the-art performance on many of the user-item data sets in the classical recommendation scenario (i.e., recommending items to the specific user), but also requires a very efficiency training and testing time, i.e., a linear time complexity in the number of items or users. It is the effectiveness and efficiency that triggers us to propose a hybrid model with latent factors and graph regularized HNM.

In this chapter, we aim to exploit the effectiveness of the latent factor models to approximate the graph linkage structures by constructing multiple profiles for each user and item. Two hybrid models are introduced in the following sections to show that with such low-dimensional latent factors we can drastically improve the computational complexity of the HNM model as well.

4.1 Latent Factor Based HNM Model (LF-HNM)

We first introduce a latent factor based model that incorporates the latent factor profiles to approximate the affinity of the user-user, user-item and item-item relationships. The motivation of such incorporation is to reduce

the computational complexity of the HNM model.

4.1.1 Latent Factor Profiles Construction

Now consider both within-domain matrices $\Phi^{(i)}$ and cross-domain matrices $\Phi^{(ij)}$, our goal is to construct latent factor profiles for each of the nodes, including users and multimedia objects.

For each of the matries, we employ PMF [10] which tries to maximize the log of the posterior distribution over the latent profiles, or equivalently, to minimize the sum-of-squared-errors objective function with quadratic regularization terms:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(\Phi_{ij} - g(U_i^{\mathsf{T}}V_j))^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N}\|U_i\|_2^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M}\|V_j\|_2^2 \quad (4.1)$$

where I_{ij} is an indicator matrix whose entries is 1 if there is an observed relations and 0 otherwise. It is worth noting that to ensure the convergence of the HNM algorithm, we map the range of prediction $U_i^{\mathsf{T}}V_j$ from $[-1, 1]$ to $[0, 1]$ with $g(x) = \frac{1}{2}(x + 1)$. This is equivalent to making prediction through

$$g(U^{\mathsf{T}}V) = \hat{U}^{\mathsf{T}}\hat{V} + \frac{1}{2}\mathbf{1}_N\mathbf{1}_M^{\mathsf{T}} \quad (4.2)$$

with $\hat{U} = \frac{1}{\sqrt{2}}U$, $\hat{V} = \frac{1}{\sqrt{2}}V$ and $\mathbf{1}_N$ is a $N \times 1$ vector of all one as the entries. Another approach is to constrain the profiles to be non-negative and we will discuss this approach in the next section.

The above optimization problem can be efficiently solved using steepest descent method. The computational complexity for this method is bounded by $\mathcal{O}(\ell b)$, where b is the number of observed relations.

Note that the factorization of each matrix is independent, therefore each node may result in multiple profiles from each factorization of related matrices. This strategy greatly reduce the complexity of the problem and hence improve the convergence of PMF, yet still maintain the accuracy of the approximation of each matrix.

4.1.2 Latent Factor Profiles in HNM Model

Now we present how to incorporate the obtained latent factor profiles to the HNM model in both the training and testing stage. We will also discuss the improvement of the computational complexities for such incorporation.

Training stage

First we consider a latent factor model for factorizing the within-domain matrices $\Phi^{(i)}$ as well as the cross-domain matrices $\Phi^{(ij)}$. Assuming the dimension of the latent factor model is ℓ, then by denoting the latent factors $p_i, q_i, r_i \in \mathbb{R}^{\ell \times 1}$, and $P^{(i)} = [p_1, \ldots, p_n]$, $Q^{(j)} = [q_1, \ldots, q_n]$, $R^{(j)} = [r_1, \ldots, r_m]$.

$$\Phi^{(i)} \approx P^{(i)\mathsf{T}} P^{(i)} + \frac{1}{2}\mathbf{1}_n \mathbf{1}_n^{\mathsf{T}} \tag{4.3}$$

and

$$\Phi^{(ij)} \approx Q^{(i)\mathsf{T}} R^{(j)} + \frac{1}{2}\mathbf{1}_n \mathbf{1}_m^{\mathsf{T}} \tag{4.4}$$

We then obtain a new update function from Eq. (3.18) by substituting the within-domain and cross-domain matrices with the latent factor matrices P and Q as following:

$$\lambda^{(i)}(f^{(i)} - y^{(i)}) + (I - P^{(i)\mathsf{T}} P^{(i)} - \frac{1}{2}\mathbf{1}_n \mathbf{1}_n^{\mathsf{T}}) f^{(i)} + \sum_{j \sim i} (Q^{(i)\mathsf{T}} R^{(j)} + \frac{1}{2}\mathbf{1}_n \mathbf{1}_m^{\mathsf{T}}) f^{(j)} = 0 \tag{4.5}$$

The main advantage of the new update function that incorporates the latent factor matrices are twofold. On one hand the latent factor models effectively estimate the overall structures. On the other hand it greatly

reduces the time complexity of our model. With $A = I + \frac{\alpha}{2-n\alpha}\mathbf{1}\mathbf{1}^\mathsf{T}$, we have

$$\begin{aligned}\Psi &= (I - \alpha\Phi)^{-1} \\ &= (I - \alpha P^\mathsf{T}P - \frac{\alpha}{2}\mathbf{1}\mathbf{1}^\mathsf{T})^{-1} \\ &= A + \alpha A P^\mathsf{T}(I - \alpha PAP^\mathsf{T})^{-1}PA \\ &= I + \frac{\alpha}{2-n\alpha}\mathbf{1}\mathbf{1}^\mathsf{T} + \alpha\hat{P}^\mathsf{T}\bar{P} \end{aligned} \tag{4.6}$$

where $\bar{P} = P + \frac{\alpha}{2-n\alpha}P\mathbf{1}\mathbf{1}^\mathsf{T} \in \mathbb{R}^{\ell\times n}$ and $\hat{P} = (I - \alpha\bar{P}P^\mathsf{T})^{-1}\bar{P} \in \mathbb{R}^{\ell\times n}$. Note it takes only $\mathcal{O}(\ell n)$ to compute $\bar{P}$ and $\mathcal{O}(\ell^2 n + \ell^3)$ to compute $\hat{P}$. Also there is no need to obtain Φ directly to do the testing, we will discuss more in the next section.

Testing stage

For the testing stage of the recommendation, we are given the initial query vector $y^{(i)}$ and need to obtain $f^{(i)}$ as the recommendation scores of each node. Following the proposed algorithm 1 we need to compute $z^{(i)}$ and $f^{(i)}$ iteratively cross each domain. As shown in Chapter 3 it takes $\mathcal{O}(n^2)$ for each iteration. In this section, we will show that with the latent factor matrices P, Q and R, we can reduce the computational complexity to $\mathcal{O}(\ell n)$, which indicates a linear complexity with respect to the number of objects to be recommended.

First we start from the computation of $z^{(i)}$:

$$\begin{aligned} z^{(i)} &= -\sum_{j=1,j\neq i}^{k} \Phi^{(ij)} f^{(j)} \\ &= -\sum_{j=1,j\neq i}^{k} Q^{(i)\mathsf{T}}\left(R^{(j)}f^{(j)}\right) + \frac{\sum f^{(j)}}{2}\mathbf{1}_n \end{aligned} \tag{4.7}$$

Hence the time complexity for computing $z^{(i)}$ is $\mathcal{O}(\ell n)$.

Table 4.1: Comparison of the proposed model and the others

Model	Information Used			Time Complexity		Handle OCCF
	ratings	social	content	training	testing	
PMF	Y	-	-	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell n)$	-
WRMF	Y	-	-	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	Y
SoRec	Y	Y	-	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell n)$	-
WRSoRec	Y	Y	-	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	Y
CTR-SMF	Y	Y	LDA	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	-
RWR	Y	Y	link-based	–	$\mathcal{O}(k^2 n^2)$	Y
GWNMTF	Y	Y	link-&sim-based	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	-
HNM	Y	Y	link-&sim-based	$\mathcal{O}(kn^3)$	$\mathcal{O}(n^2)$	Y
LF-HNM	Y	Y	link-&sim-based	$\mathcal{O}(k^2\ell^2 n)$	$\mathcal{O}(\ell n)$	Y

With the obtained $z^{(i)}$ we can now compute $f^{(i)}$ by:

$$\begin{aligned} f^{(i)} &= \Psi\left((1-\alpha)y^{(i)} + \alpha z^{(i)}\right) \\ &= (I + \alpha\hat{P}^{\mathsf{T}}\bar{P})\left((1-\alpha)y^{(i)} + \alpha z^{(i)}\right) \\ &= (1-\alpha)y^{(i)} + \alpha z^{(i)} + \alpha\hat{P}^{(i)\mathsf{T}}\bar{P}^{(i)}\left((1-\alpha)y^{(i)} + \alpha z^{(i)}\right) \end{aligned} \tag{4.8}$$

Therefore it also takes $\mathcal{O}(\ell n)$ to compute $f^{(i)}$. Table 4.1 shows the comparison of the computational complexity of the hybrid model to the other models.

4.1.3 Experimental Results

We conduct the experiments on the same *Flickrgroup* data set as in Chapter 3.5. A 50-50 training/testing users split is employed which results in 9,699 images and 20,298 users with around 158,000 favor links. The 1000 query users are randomly selected out of 20,298 users. The dimension of the latent factors is set to one of {1,10,30,100,300}. We compare the performance of LF-HNM to the other recommendation models in terms of mAPS (in Table 4.2) and running time (in Table 4.3). As depicted in Figure 4.1, the performance of the hybrid LF-HNM method outperforms all the other models, including HNM. This indicates that the latent factors model indeed captures the intrinsic structure of the within-domain and cross-domain matrices and provides extra information during the recommendation process.

We further provide the mAPS from different settings of dimensionality of

Table 4.2: Performance comparison on *Flickrgroup* data set with different methods

	mAPS	Prec@1	Prec@5	Prec@10	Prec@20
WRMF	38.42	0.0160	0.0124	0.0097	0.0077
WRSoRec	34.12	0.0280	0.0182	0.0141	0.0104
RWR	36.18	0.0130	0.0116	0.0083	0.0066
HNM	27.98	**0.0440**	0.0220	0.0157	0.0118
LF-HNM	**25.29**	0.0410	**0.0268**	**0.0201**	**0.0155**

the latent factors in Table 4.4. When the latent factor is set to 1-D, the mAPS is a worst 48.38 that indicates the latent factors are not able to make close estimation of Φ or is even making incorrect estimations on the unobserved relations. On the other hand, the mAPS from the 100-D or 300-D latent factors is slightly worse than that from 30-D latent factor, which may be due to the (slightly) overfitting during the matrix factorization.

From Table 4.3, it is evident that the computational complexity is greatly reduced. A further breakdown in training time can be found in Table 4.4, which shows the training time for the latent factor matrix factorization and the training time of obtaining $\hat{P}$. One can see that the time needed to compute $\hat{P}$ is almost ignorable comparing to the time needed for factorization. In terms of the testing time, with the hybrid model, it takes 0.0086 seconds for making recommendations to the user, improving from 0.2612 seconds. All experiments were performed on a machine equipped with an Intel Core i5 2.3GHz CPU (Dual Core) and 8 GB memory.

Table 4.3: Comparison of training and testing time used (in seconds)

	Training	Testing
WRMF	109	0.3
WRSoRec	462	0.3
RWR	-	1384.7
HNM	1536	261.2
LF-HNM	455	8.6

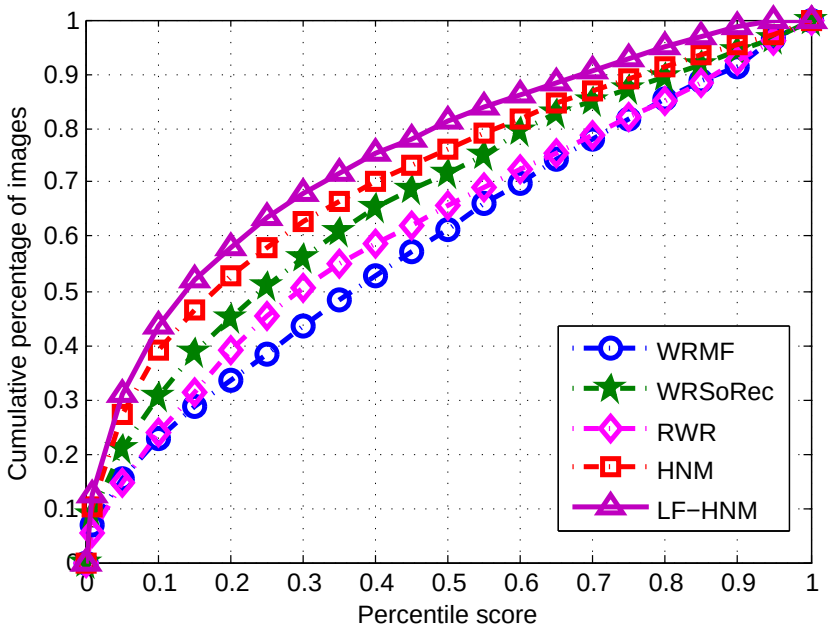


Figure 4.1: P-PS curves on image recommendation to specific users by five models

Table 4.4: Performance of LF-HNM with different dimension of latent factors (time in seconds)

# Latent factor	1	10	30	100	300
mAPS	48.38	29.06	25.29	27.12	27.98
Training time (PMF)	385	426	455	481	566
Training time ($\bar{P}$ and $\hat{P}$)	0.001	0.023	0.105	0.461	1.712
Testing time	0.71	3.53	8.61	19.32	46.06

4.2 HNM Regularized LF Model (HNM-LF)

In the previous section we introduced an hybrid model, latent factor based HNM model (LF-HNM), which adopts the users' and items' latent factors (profiles) to approximate the smoothness matrices. In this section, we will introduce another hybrid model that makes use of the smoothness matrices from the HNM regularization model to construct the users' and items' profiles. Notations used in this section are summarized in Table 4.5.

Table 4.5: Notations

Notation	Description
N, M	number of users (N) and items (M)
R, I	rating matrix (R) and indicating matrix for observed ratings (I)
U, V	profiles for users (U) and items (V)
ℓ	dimensionality of the profiles
Φ	smoothness matrix due to HNM regularization

4.2.1 Non-Negative Latent Factor Profiles Construction with HNM Regularization

Weighted non-negative matrix factorization (WNMF) has been widely used to attack the non-negative matrix completion problems, including rating prediction for recommendation [44]. There are several extensions of WNMF that try to incorporate side information to improve the recommendation accuracy. Among them, graph regularized weighted non-negative matrix factorization (GWNMF) [37] is the state-of-the-art method which adds regularization constraints on the user and item graph so the profiles of similar users/items are similar. However, one of the main issues with GWNMF is that it does not consider the heterogeneous information. Therefore, in this section we will extend GWNMF with our proposed HNM regularization to construct the profiles such that not only profiles of similar objects of the same domain are similar, but profiles of similar (similar in the HNM sense) objects across different domains are also similar.

Consider a non-negative rating matrix $R \in \mathbb{R}_+^{N \times M}$, we are interested in finding ℓ-dimensional non-negative profiles of users ($U \in \mathbb{R}_+^{\ell \times N}$) and items ($V \in \mathbb{R}_+^{\ell \times M}$) such that the inner product of the profiles recovers the ratings. To achieve the goal, we formulate the following objective function, which is often named WNMF:

$$\min_{U,V} F(U, V; R) = \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^\mathsf{T} V_j)^2$$
$$\text{subject to } U \geq 0, V \geq 0 \tag{4.9}$$

where U_i is a ℓ-dimensional column vector as the i-th user's profile and V_j is a ℓ-dimensional column vector as the j-th item's profile.

We extend the WNMF model by adding the HNM regularization on the

latent factors to Eq. (4.9), making it become

$$\begin{aligned} &\min_{U,V} C(U,V;R,\Phi) = F(U,V;R) + G(U,V;\Phi) \\ &\text{subject to } U \geq 0, V \geq 0 \end{aligned} \tag{4.10}$$

where $\geq$ indicates the element-wise non-negativity for the matrix U and V, and $G(U,V;\Phi)$ is the HNM regularization to the latent factors based on the smoothness matrices from the HNM model. In particular, consider the within-domain smoothness matrix and cross-domain smoothness matrix we can write $G(U,V;\Phi)$ as follows:

$$\begin{aligned} G(U,V;\Phi) = &\lambda_U \text{tr}(U\Phi^{(UU)}U^\mathsf{T}) + \lambda_V \text{tr}(V\Phi^{(VV)}V^\mathsf{T}) \\ &+ \lambda_{UV}\text{tr}(U\Phi^{(UV)}V^\mathsf{T}) + \lambda_{VU}\text{tr}(V\Phi^{(VU)}U^\mathsf{T}) \end{aligned} \tag{4.11}$$

Note that Eq. (4.11) can be written for the more general case where more than two domains are involved:

$$G(U_1, U_2, \ldots, U_p; \Phi) = \sum_{i=1}^{p}\sum_{j=1}^{p} \lambda_{ij}\text{tr}(U_i\Phi^{(ij)}U_j^\mathsf{T}) \tag{4.12}$$

Also, the state-of-the-art graph regularized matrix factorization method (GWNMF) [37] is indeed a special case of our proposed HNM regularized model in Eq. (4.12), where $p = 2$ and $\lambda_{ij} = 0$ for $i \neq j$.

Now we go back to the two domain case where we consider only users and items for now. To solve the optimization problem in Eq. (4.10) we follow the approach for solving the WNMF problem where an alternating scheme on solving subproblems with U and V is used.

Solution of subproblem w.r.t U

The subproblem of Eq. (4.10) with respect to U can be formulated as follows:

$$\begin{aligned} &\min_{U} C(U;R,V,\Phi) = F(U;R,V) + G(U;V,\Phi) \\ &\text{subject to } U \geq 0 \end{aligned} \tag{4.13}$$

where

$$F(U;R,V) = \sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^{\mathsf{T}}V_j)^2$$
$$G(U;V,\Phi) = \lambda_U \mathrm{tr}(U\Phi^{(UU)}U^{\mathsf{T}}) + \lambda_{UV}\mathrm{tr}(U\Phi^{(UV)}V^{\mathsf{T}}) + \lambda_{VU}\mathrm{tr}(V\Phi^{(VU)}U^{\mathsf{T}})$$

From the Karush-Kuhn-Tucker (KKT) complementary condition on the non-negativity of U we have

$$U_{ij}\left[\frac{\partial\, C(U)}{\partial\, U}\right]_{ij} = 0 \tag{4.14}$$

which gives

$$U_{ij}(I_{ij}V_{i.}V^{\mathsf{T}}U_{.j} - I_{ij}V_{i.}R_{j.}^{\mathsf{T}} + \lambda_U U_{i.}\Phi_{.j}^{(UU)} + \frac{1}{2}\lambda_{UV}U_{i.}\Phi_{.j}^{(UV)} + \frac{1}{2}\lambda_{VU}V_{i.}\Phi_{.j}^{(VU)}) = 0 \tag{4.15}$$

where subscript $i.$ indicates the i-th row of the matrix and $.j$ indicates the j-th column of the matrix. Since the smoothness matrix Φ may take negative values, we decompose it into two parts: positive part $\hat{\Phi} = (\mathrm{abs}(\Phi) + \Phi)/2$ and negative part $\bar{\Phi} = (\mathrm{abs}(\Phi) - \Phi)/2$. The function $\mathrm{abs}(\Phi)$ indicates taking absolute values of all the entries in Φ. Note the $\Phi = \hat{\Phi} - \bar{\Phi}$. We then substitute $\hat{\Phi}$ and $\bar{\Phi}$ into Eq. (4.15) and obtain the updating formula:

$$U_{ij} \leftarrow U_{ij}\frac{I_{ij}V_{i.}R_{j.}^{\mathsf{T}} + \lambda_U U_{i.}\bar{\Phi}_{.j}^{(UU)} + \frac{1}{2}\lambda_{UV}U_{i.}\bar{\Phi}_{.j}^{(UV)} + \frac{1}{2}\lambda_{VU}V_{i.}\bar{\Phi}_{.j}^{(VU)}}{I_{ij}V_{i.}V^{\mathsf{T}}U_{.j} + \lambda_U U_{i.}\hat{\Phi}_{.j}^{(UU)} + \frac{1}{2}\lambda_{UV}U_{i.}\hat{\Phi}_{.j}^{(UV)} + \frac{1}{2}\lambda_{VU}V_{i.}\hat{\Phi}_{.j}^{(VU)}} \tag{4.16}$$

Solution of subproblem w.r.t V

Similar to U, we have the subproblem of Eq. (4.10) with respect to V formulated as follows:

$$\min_V C(V;R,U,\Phi) = F(V;R,U) + G(V;U,\Phi)$$
$$\text{subject to } V \geq 0 \tag{4.17}$$

where

$$F(V;R,U) = \sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^{\mathsf{T}}V_j)^2$$
$$G(V;U,\Phi) = \lambda_V \mathrm{tr}(V\Phi^{(VV)}V^{\mathsf{T}}) + \lambda_{UV}\mathrm{tr}(U\Phi^{(UV)}V^{\mathsf{T}}) + \lambda_{VU}\mathrm{tr}(V\Phi^{(VU)}U^{\mathsf{T}})$$

From the Karush-Kuhn-Tucker (KKT) complementary condition on the non-negativity of V we have

$$V_{ij}\left[\frac{\partial\, C(V)}{\partial\, V}\right]_{ij} = 0 \tag{4.18}$$

which gives

$$V_{ij}(I_{ij}U_{i.}U^{\mathsf{T}}V_{.j} - I_{ij}U_{i.}R_{.j} + \lambda_V V_{i.}\Phi_{.j}^{(VV)} + \frac{1}{2}\lambda_{UV}U_{i.}\Phi_{.j}^{(UV)} + \frac{1}{2}\lambda_{VU}V_{i.}\Phi_{.j}^{(VU)}) = 0 \tag{4.19}$$

where subscript $i.$ indicates the i-th row of the matrix and $.j$ indicates the j-th column of the matrix. Since the smoothness matrix Φ may take negative values, we decompose it into two parts: positive part $\hat{\Phi} = (\mathrm{abs}(\Phi) + \Phi)/2$ and negative part $\bar{\Phi} = (\mathrm{abs}(\Phi) - \Phi)/2$. The function $\mathrm{abs}(\Phi)$ indicates taking absolute values of all the entries in Φ. Note the $\Phi = \hat{\Phi} + \bar{\Phi}$. We then substitute $\hat{\Phi}$ and $\bar{\Phi}$ into Eq. (4.19) and obtain the updating formula:

$$V_{ij} \leftarrow V_{ij}\frac{I_{ij}V_{i.}R_{.j} + \lambda_V V_{i.}\bar{\Phi}_{.j}^{(VV)} + \frac{1}{2}\lambda_{UV}U_{i.}\bar{\Phi}_{.j}^{(UV)} + \frac{1}{2}\lambda_{VU}V_{i.}\bar{\Phi}_{.j}^{(VU)}}{I_{ij}U_{i.}U^{\mathsf{T}}V_{.j} + \lambda_V V_{i.}\hat{\Phi}_{.j}^{(VV)} + \frac{1}{2}\lambda_{UV}U_{i.}\hat{\Phi}_{.j}^{(UV)} + \frac{1}{2}\lambda_{VU}V_{i.}\hat{\Phi}_{.j}^{(VU)}} \tag{4.20}$$

4.2.2 Complexity Analysis

Table 4.6 again lists the comparison between all the proposed HNM related methods and the baseline methods, including their variants for dealing with the one-class collaborative filtering problem. As we can see in this table, the newly proposed HNM-LF requires a higher computational complexity for the training stage comparing to the first hybrid model LF-HNM proposed in Section 4.1. On the other hand, because the HNM-LF method is to construct the profiles for both users and items, the testing stage would need only an inner product of the two profiles, so it takes the same complexity as that of

the latent factor based methods such as PMF and SoRec.

Table 4.6: Comparison of the proposed model and the others

Model	Information Used			Time Complexity		Handle OCCF
	ratings	social	content	training	testing	
PMF	Y	-	-	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell n)$	-
WRMF	Y	-	-	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	Y
SoRec	Y	Y	-	$\mathcal{O}(\ell n)$	$\mathcal{O}(\ell n)$	-
WRSoRec	Y	Y	-	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	Y
CTR-SMF	Y	Y	LDA	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	-
RWR	Y	Y	link-based	–	$\mathcal{O}(k^2 n^2)$	Y
GWNMTF	Y	Y	link-&sim-based	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	-
HNM	Y	Y	link-&sim-based	$\mathcal{O}(kn^3)$	$\mathcal{O}(n^2)$	Y
LF-HNM	Y	Y	link-&sim-based	$\mathcal{O}(k^2\ell^2 n)$	$\mathcal{O}(\ell n)$	Y
HNM-LF	Y	Y	link-&sim-based	$\mathcal{O}(\ell n^2)$	$\mathcal{O}(\ell n)$	-

4.2.3 Experimental Results

In this section, we conduct experiments on the *Movielens100K* data set to compare the proposed HNM-LF method with some state-of-the-art methods, e.g., PMF, SoRec, RWR, GWNMTF, HNM and LF-HNM introduced in previous section. Note that we do not do experiment on *Flickrgroup* data set as the HNM-LF is not designed to handle the OCCF problem.

Movielens100K

For the *Movielens100K* data set, we use an 80-20 training/testing rating split on the 100 K movie ratings of 943 users for 1682 movies. The weight of the user-user relationship is based on a 30-dimensional binary vector from the users' demographical information, namely age, gender and occupation. In particular, 6 out of 29 dimensions are used to indicate the age group that the user belongs to: {1-17, 18-24, 25-34, 35-44, 45-49, 50+}. And 2 out of 29 dimensions are to indicate the gender of the user. The rest of the 21 dimensions are used to indicate the occupation of the user. If two users are of the same age group, the weight will increase by 1/3; and if two users are of the same gender, the weight will increase by another 1/3; and if two users are of the same occupation, the weight will increase by another 1/3. The minimum

of the weight between two users is 0 when two users are not in the same age group and of not the same gender nor the occupation. The maximum of the weight between two users is 1 when two users are of the same age range, gender and occupation. The weight of the item-item relationship is based on the movie genres the item belongs to. There are in total 19 movie genres and one movie at least belongs to one of these 19 genres. Each item is represented by a 19-dimensional binary vector where each entry of the vector indicates whether the item belongs to the genre. The weight between two items is then the cosine similarity of the two vectors of the items. Note that the minimum weight of the items is also 0. For the weight of user-item relationship, we simply use the normalized ratings where the ratings are scaled by 1/5.

We evaluate the performance based on both the rating prediction (MAE) and the ranking (nDCG). The rating prediction is evaluated with mean absolute error (MAE) which is defined as:

$$MAE = \frac{\sum_{i,j} I_{ij} * |R_{ij} - \hat{R}_{ij}|}{\sum_{i,j} I_{ij}} \tag{4.21}$$

where $\hat{R}_{ij}$ is the predicted rating that i-th user would give j-th item and R_{ij} is its ground truth value. Note that the best MAE is 0 and the lower MAE the better performance.

For the ranking based evaluation, because the ratings are of values from 1-5, we did not use the mAPS metric but used the popular nDCG evaluation metric that have been widely used in the ranking task. The normalized discounted cumulative gain (nDCG) metric for a list of p items ,V, is defined as follows:

$$\begin{aligned} \text{nDCG}_p(V) &= \frac{\text{DCG}_p(V)}{\text{IDCG}_p(V)} \\ \text{DCG}_p(V) &= \sum_{i=1}^{p} \frac{2^{rel(i)} - 1}{log_2(i+1)} \\ \text{IDCG}_p(V) &= \text{DCG}_p(\hat{V}) \end{aligned} \tag{4.22}$$

$\hat{V}$ is the ideal ordering of the list V based on the relevance of the items. Note that the best nDCG is 1 and the higher nDCG the better performance. For this evaluation, we report $p \in \{1, 10, 20, 100, all\}$ (all means to use all the 1682 movies).

The settings of the experiment are as follows: the 100 K ratings are equally divided into five parts. With these five parts we do fivefold cross validation so that each of the five evaluations has 80 K ratings for training and 20 K ratings for testing. The performance was reported by the average of the five evaluations. We compare the performance of seven methods, including four baseline methods and three proposed methods. The parameter settings for each method are as follows:

- *PMF* [10]: the dimension of the latent factors is set to {5,10,20} and the regularization parameter is set to {0.01, 0.1, 1}.
- *SoRec* [26]: the dimension of the latent factors is set to {5,10,20} and the regularization parameter is set to {0.001, 0.01, 0.1, 1}.
- *RWR* [21]: the damping parameter is set to {0.1, 0.3, 0.5, 0.7, 0.9}.
- *GWNMTF* [37]: the dimension of the latent factors is set to {5,10,20} and the two regularization parameters are set equally to {0.001, 0.01, 0.1, 1}.
- *HNM*: the two regularization parameters are set equally to {0.0001, 0.001, 0.01, 0.1,1}.
- *LF-HNM*: the dimension of the latent factors is set to {5,10} and the two regularization parameters are set equally to {0.0001, 0.001, 0.01, 0.1}.
- *HNM-LF*: the dimension of the latent factors is set to {5,10,20} and the two regularization parameters are set equally to {0.001, 0.01, 0.1, 1}.

We perform the grid evaluation to find the best parameter settings for each method and report the performance with those settings.

Table 4.7 summarizes the performance of the baseline methods and the proposed methods in terms of MAE and nDCG. First we notice that the performance of HNM regularized LF method (HNM-LF) achieves the best performance in both metrics. Second, RWR, HNM and LF-HNM do not perform good in terms of MAE because they are not designed to predict the exact ratings but to find the ranking of the items. These three methods do outperform all the baseline methods in terms of ranking metric nDCG. We

Table 4.7: Comparison of MAE and nDCG on *MovieLens100K* data set

	MAE	nDCG	nDCG1	nDCG10	nDCG20	nDCG100
PMF	0.7577	0.7318	0.8605	0.7596	0.742	0.7224
SoRec	0.7501	0.747	0.8569	0.7783	0.7732	0.7497
RWR	0.8213	0.8304	0.8482	0.8425	0.8392	0.8439
GWNMTF	0.7179	0.8511	0.8659	0.8778	0.8715	0.8626
HNM	0.7925	0.8719	0.9124	0.883	0.8825	0.8835
LF-HNM	0.7811	0.8731	0.9147	0.8851	0.8845	0.8853
HNM-LF	**0.7154**	**0.8845**	**0.9305**	**0.8996**	**0.8942**	**0.8969**

also note that the state-of-the-art method GWNMTF does perform fairly well for both metrics, but its ignorance on the cross-domain information make its performance not as good as our HNM-LF model which considers all the available information.

On the other hand, Table 4.8 reports the training and testing time in seconds for evaluating the *MovieLens100K* data set. It is clear that the proposed HNM-LF method as a latent factor based method enjoys its efficiency in the testing phase.

Table 4.8: Comparison of training and testing time used on *MovieLens100K* data set (in seconds)

	Training	Testing
PMF [10]	34.19	0.01
SoRec [26]	70.67	0.01
RWR [21]	-	491.37
GWNMTF [37]	177.85	0.03
HNM	528.10	67.88
LF-HNM	82.94	3.12
HNM-LF	155.12	0.02

CHAPTER 5

RECOMMENDATION SCENARIOS

So far we have discussed about the classical recommendation task, that is, to recommend items to users based on their preference history as well as the social activities and relationships and the content information. Nevertheless, there are still a variety of recommendation scenarios that a modern recommender system may encounter.

In this chapter, we would like to study several such scenarios, including context-specific recommendations in Section 5.1, cold-start recommendations or new user onboarding in Section 5.2, and preference drifting in Section 5.3. In each of the section, we aim to describe how to adapt our proposed model to handle the observed pattern for the specific scenario.

5.1 Context-Specific Recommendations

Traditional recommender systems make the item recommendations to users without any contextual specificity. Yet recent boosts of social media sites such as Flickr or YouTube contain multimedia objects, which occur in the context of an extensive amount of content information such as tags, image content, in addition to the user preferences for the different objects. Such rich information brings an important desiderata for the recommender systems: context-specific recommendation. Specifically, the recommended results are corresponded to a particular subject, as may be specified by a particular user. For example, in the context of an image media network, it may be possible to focus the recommendation results in terms of different keywords or image targets. This leads to recommendations with varying levels of specificity, as follows:

- For a given user X, determine the highest ranked image recommendations related to the keyword "northern lights".

- For a given user X, a target image Y, and the keyword "northern lights" determine the highest ranked image recommendations related to the target image Y and the keyword "northern lights".
- For a given image keyword "northern lights", determine the most likely users to be interested in that subject.
- For a given target image Y, determine the most likely users to be interested in it.

The first two kinds of recommendations are useful for recommending context-specific recommendations to a specific user, whereas the next two kinds of recommendations may be useful for determining individuals with interests in specific kinds of content. We note that the nature of the recommendation contexts are quite complex, in this case, and require knowledge of the attribute and linkage information related to the objects. Clearly, straightforward collaborative filtering methods [1, 2, 3] cannot be used directly in such scenarios because of the complexity of finding recommendations related to specific kinds of content.

5.1.1 Context-Specific Recommendation with HNM

One advantage of the proposed HNM model is the ease to incorporate the context specificities in the recommendation decision process.

Recall that the target vector $y^{(i)}$ for domain i, which is formulated as follows:

$$y^{(i)} = \xi \frac{1}{N}\mathbf{1} + (1-\xi)q^{(i)} \tag{5.1}$$

We noted in Chapter 3 that for the traditional recommendation scenario, i.e., to recommend items to a specific user, the entries in $q^{(i)}$ can be determined as:

$$q^{(i)}(v) = \begin{cases} 1, & \text{if } i \text{ is the user domain and} \\ & \quad v \text{ corresponds to the specific query user;} \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

To enable the context-specific recommendations, one can define the

following target vector to realize the contextual components requested by the users.

$$q^{(i)}(v) = \begin{cases} \frac{1}{N_w} w_q^{(i)}(v), & \text{if } v \text{ corresponds to the requested} \\ & \quad \text{contextual components;} \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

where N_w is a normalization term to make sure that $\sum_{i \in \mathcal{D}} \sum_{v \in V^{(i)}} q^{(i)}(v) = 1$.

In some cases there may not exist exact corresponding components in the network. In such cases, it would be useful to incorporate the top-K nearest neighbors that best match the requested contextual components based on the intra-domain content similarities.

$$q^{(i)}(v) = \begin{cases} \frac{1}{N_w} Sim^{(i)}(v, u), & \text{if } v \text{ is the kNN components to the} \\ & \quad \text{requested contextual components } u; \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

5.1.2 Experiment Settings

For the experiments regarding *contextual recommendations*, we only compare our results with a slightly modified version of RWR [21] but not PMF nor SoRec because these two method do not incorporate any content information. The way we modify RWR algorithm is to specify the restart vector based on the requested context. Since the context was always picked from among the existing nodes in the network, it was easy to modify the algorithm, in which restart was also allowed at the context node with an equal probability. For example, in order to determine recommendations for a user X in the context of image Y, we allowed a restart at either of these nodes with equal probability. Similarly, in order to determine recommendations for a user X in the context of keyword, which also occurs as a tag node Z, we allowed a restart at either of these nodes with equal probability. Again, all experiments were performed on a machine equipped with an Intel Core i5 2.3 GHz CPU (Dual Core) and 8 GB memory.

5.1.3 Image Recommendation for Specific Users with Image Context

We used the same setting as the first experiment, except that we added a randomly chosen image from the training data as *image context* for the query. The proposed HNM and LF-HNM method again outperform the RWR method by 13.6% and 21.3%, respectively (from 35.95 to 31.07 and 28.28, respectively). Figure 5.1 illustrates the P-PS curves with the proposed method and the baseline method for this testing scenario. It clearly shows that the proposed method gives better recommendations over the baseline method. On the other hand, we notice that the mAPS of this scenario is slightly higher than the first scenario, indicating the first scenario is easier than this one. It is not surprising as the given image query serves as a strong constraint to the recommendation problem. Figure 5.2 shows the recommended images for a specific user. The provided training images favored by the user are listed in the top row, and the query image is at the upper-right corner. The middle block shows the top 10 recommended images with the proposed algorithm, and the bottom block shows the top 10 recommended images with the baseline algorithm. The images with the green circle are truly favored by the user, according to the ground truth "like" links (removed for testing purpose).

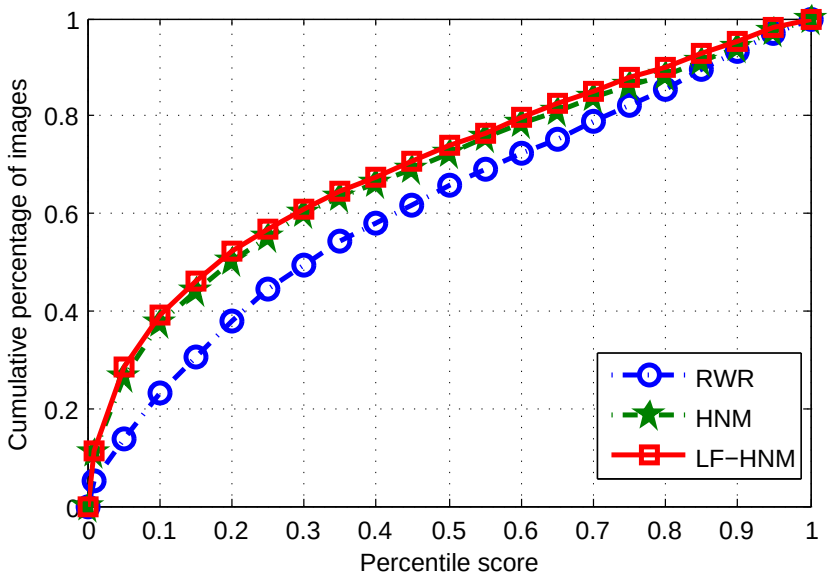


Figure 5.1: P-PS curve for image recommendation with image context

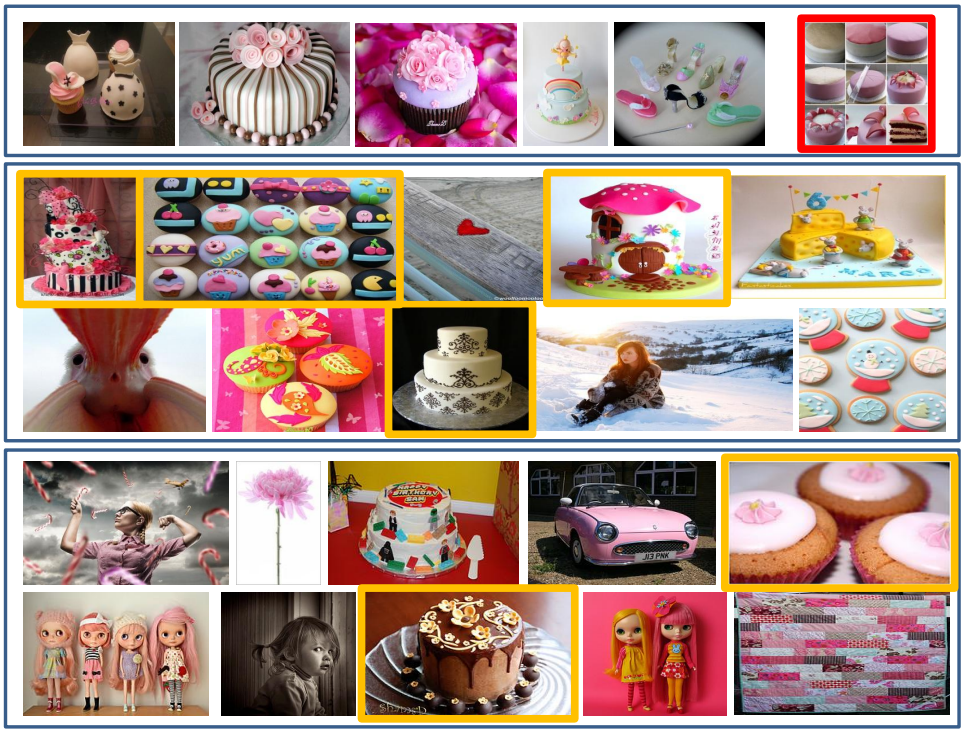


Figure 5.2: Top 10 recommended images for a given user based on the given query image (with a red bounding box); top row shows the training images favored the given user, the middle block shows the results from proposed method and the bottom block shows the results from baseline method; images with orange circles are the testing images truly favored by the user

5.1.4 Image Recommendation for Specific Users with Keyword Context

Same setting is used as in this first experiment, except one text word associated with a randomly selected image is picked as the query text. The mean average percentile score for the baseline RWR algorithm is 32.45 while the proposed HNM and LF-HNM algorithm has a score of 30.13 and 27.25. The P-PS curves in Figure 5.3 also supports this conclusion. Figure 5.4 illustrates the recommended images for a specific user, who provided the query texts "automotive" and "car", which are selected from the texts associated with the training images that this user favors. The images in the top block are those the user favors and the images in the middle block are the recommendations from the proposed method while the bottom block provides the recommended images using the baseline method.

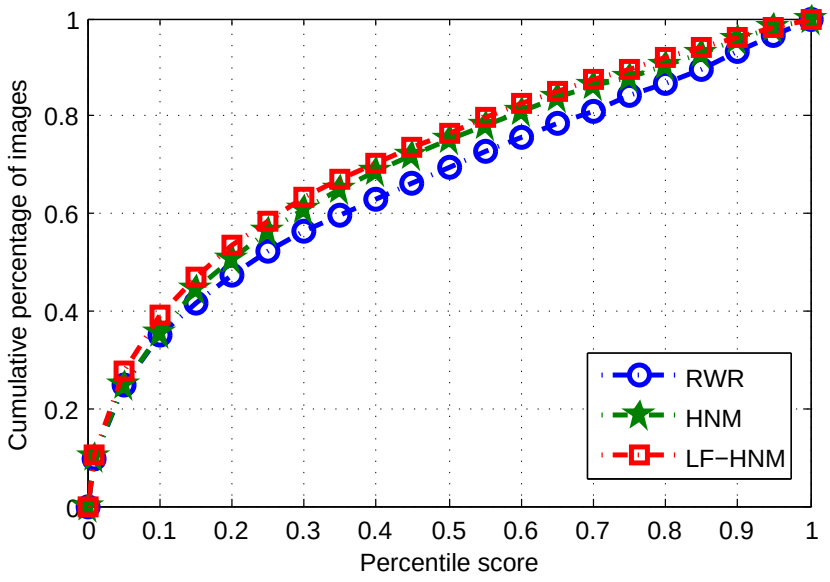


Figure 5.3: P-PS curve for image recommendation with keyword context

5.1.5 Image Recommendation for Specific User Group

There is also an interesting and practical scenario where we want to recommend items to a group of users. This group of users may share similar interests (e.g., club members or product fans), or they may not share similar interests (e.g., family members). The latter scenario is more challenging so in this section we will focus more on the first scenario. As mentioned earlier, users in Flickr can join groups which have specific interests toward the content of the images. Figure 5.5 illustrates some examples of the user groups and their favored images, called *group pool*, in Flickr.

To evaluate the performance of an image recommendation to user groups, we further collect the the images from the group pool of the 139 user groups, resulting in more than 118,000 images. We note that these 118,000 images are not a subset of 2.2 million images that the group members favored, nor a superset of the 9,699 images used in the *Flickrgroup* data set. The main reason is that the group members usually have multiple interests so their favored images are usually very diversified and only part of them would be related to the group pool. We then pick 100 user groups whose group pool has more than 300 images for evaluation. Out of the 300 images, 200 images are used as training images and 100 images are used as testing images.

We conduct the experiments in two settings: the first one is to use purely

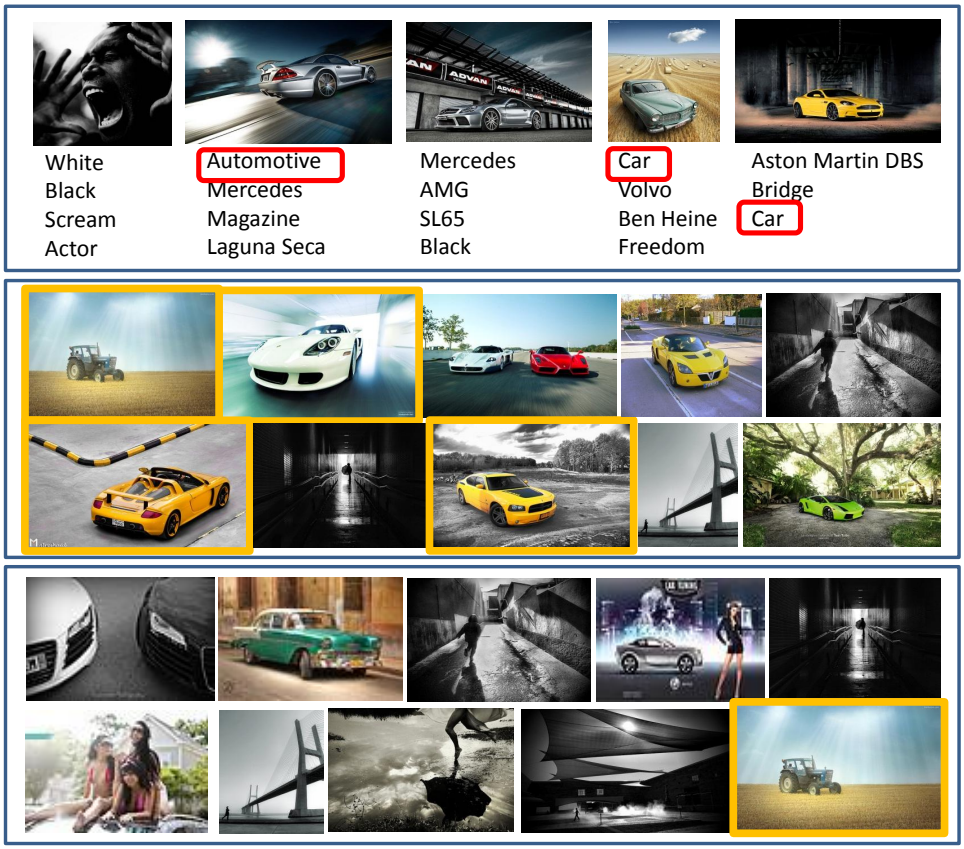


Figure 5.4: Top 10 recommended images for a given user based on query keywords "automotive" and "car" circled in red; top row shows the training images favored by user, the middle block shows the results from proposed method and the bottom block shows the results from baseline method; images with orange circles are the truly favored test images (ground truth)

the group pool information, that is, we take the preferences of the 100 groups on those 30,000 images as the input. The group-group relationships are constructed based on their group members, so if two groups have many overlapping members then these two groups are considered more similar. The second setting is to use the user preferences plus the group preferences. So there are in total 39,699 images. The preferences of the groups are expanded to their group members based on the group-user relationships such that the group members are assumed to like all the images in the group pool. Since there is no longer the entity of group involved in this setting, we must apply context-specific recommendations to the setting so that we can make recommendations to the group based on its group

Figure 5.5: Examples of user groups and their favored images in Flickr

members.

Figure 5.6 illustrates the mAPS performance of the recommendations to the 100 Flickr groups. First we observe that the WRMF and WRSoRec methods do not work on the second setting because they do not realize the context-specific recommendation. Also, the LF-HNM method does not perform well compared to HNM in the first setting, mainly because its approximate smoothness matrix from WRMF is poor. Furthermore, we note for the methods that work for both settings, their performance for the second setting is much better than the first setting. This is due to the incorporation of the additional user preference provide helpful information. However, the second setting is only possible with the context-specific recommendations so the recommendations can be made to the groups.

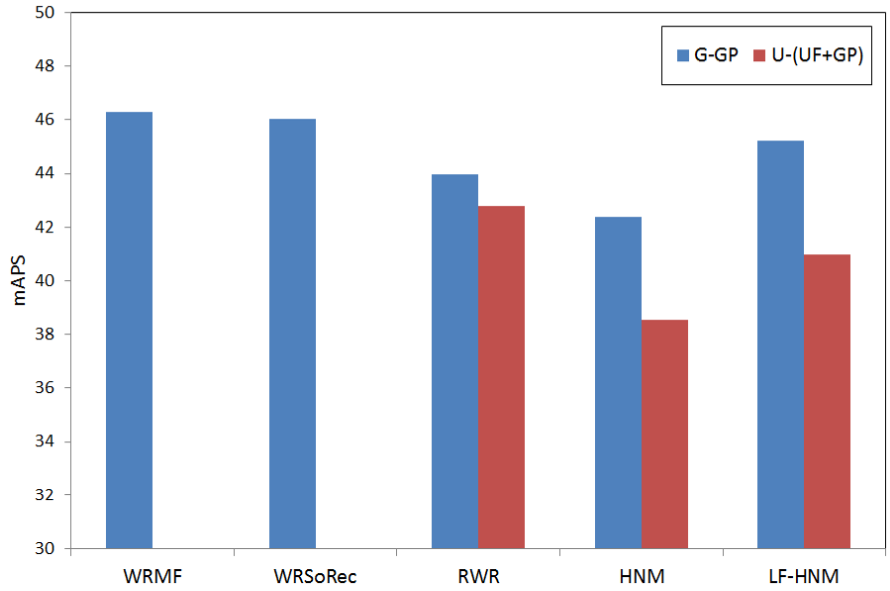


Figure 5.6: Performance comparison for recommendations to a group of user

5.2 New User Onboarding

Making recommendations to a new user who just joins the system is probably one of the most challenging recommending tasks. Usually the system has to design a serial of questionnaires to guide the new users to the items they are interested in. However, such a questionnaire design is not trivial and it requires extra effort from the users. Gladly, with the advent of online social networks, new users' social relations are likely available and accessible before their first request to the recommender system. For example, a new user to YouTube may already have his/her social relations available in the Google+ or GMail, and the first thing new users to Facebook will do is most likely to find their friends. Therefore, it is critical that the recommender system can exploit the social information from new users and quickly update the recommendation models to make personalized recommendations to the new users based on their social information.

5.2.1 HNM Model: Fast Update for New Users

Let $\phi \in \mathbb{R}^n$ denote the social relationships of a new user to the n existing users in the system. Note that in the real-world scenario ϕ is going to be a sparse vector. Without loss of generality, we assume the entries in ϕ are

non-negative numbers, meaning only positive relationships (e.g., friendships) and no negative relationships (e.g., enemies) are considered, also the relationships are equally weighted. With such assumptions, we can write $\phi = \kappa[\phi_1,\ \phi_2, \ldots,\ \phi_n], \phi_i \in \{0, 1\}$ where κ is a positive number. Also, we use r to denote the number of non-zero entries in ϕ so $r = \|\phi\|_0$.

Our model before considering the new user is:

$$\mathcal{L} \equiv (D - \alpha W)^{-1} \tag{5.5}$$

$$D = \texttt{diag}(D_i), D_i = d(i) \tag{5.6}$$

$$W_{ij} = w(i, j) \tag{5.7}$$

Now the update graph with the new user has new weight matrices as:

$$D' = \texttt{diag}(D'_i), D'_i = d(i) + \phi_i, D'_{i+1} = \sum_i \phi_i \tag{5.8}$$

$$W' = \begin{pmatrix} W & \phi \\ \phi^T & 0 \end{pmatrix} \tag{5.9}$$

And it can be shown that

$$\begin{aligned} \mathcal{L}' &= (D' - \alpha W')^{-1} \\ &= \begin{pmatrix} D + \texttt{diag}(\phi) - \alpha W & -\alpha\phi \\ -\alpha\phi^T & \sum_i \phi_i \end{pmatrix}^{-1} \\ &= \begin{pmatrix} \bar{\mathcal{L}} + \eta\alpha^2 \bar{\mathcal{L}}\phi\phi^T\bar{\mathcal{L}} & -\eta\alpha\bar{\mathcal{L}}\phi \\ -\eta\alpha\phi^T\bar{\mathcal{L}} & \eta \end{pmatrix} \end{aligned} \tag{5.10}$$

where $\bar{\mathcal{L}} = \mathcal{L} + \mathcal{L}_{.\phi}(I - \mathcal{L}_{\phi\phi})^{-1}\mathcal{L}_{\phi.}$ and $\eta = \frac{1}{\sum_i \phi_i - \alpha^2\phi^T\mathcal{L}\phi}$. $\mathcal{L}_{.\phi}$ is an n-by-r matrix that keeps columns of $\mathcal{L}$ based on ϕ and similarly, $\mathcal{L}_{\phi\phi}$ is an r-by-r matrix that keeps columns and rows of $\mathcal{L}$ based on ϕ. Note that computing $\bar{\mathcal{L}}$ takes $\mathcal{O}(rn^2)$, hence it takes $\mathcal{O}(rn^2)$ to update the model with a new user.

Proof of Eq. (5.10) : First we will show that $\bar{\mathcal{L}} = (D + \texttt{diag}(\phi) - \alpha W)^{-1} = \mathcal{L} + \mathcal{L}_{.\phi}(I - \mathcal{L}_{\phi\phi})^{-1}\mathcal{L}_{\phi.}$.

Let $(D - \alpha W) = V\Sigma V^T$, then

$$\begin{aligned}(D + \texttt{diag}(\phi) - \alpha W) &= V\Sigma V^T + \texttt{diag}(\phi) \\ &= V(\Sigma + V^T\texttt{diag}(\phi)V)V^T \\ &= V(\Sigma + V_\phi^T V_\phi)V^T \end{aligned} \tag{5.11}$$

where $V_\phi = \texttt{diag}(\phi)^{\frac{1}{2}}V$. Therefore, with $\mathcal{L} = (D - \alpha W)^{-1} = V\Sigma^{-1}V^T$, we have

$$\begin{aligned}\bar{\mathcal{L}} &= (D + \texttt{diag}(\phi) - \alpha W)^{-1} \\ &= V(\Sigma + V_\phi^T V_\phi)^{-1}V^T \\ &= V\left(\Sigma^{-1} + \Sigma^{-1}V_\phi^T(I - V_\phi\Sigma^{-1}V_\phi^T)^{-1}V_\phi\Sigma^{-1}\right)V^T \\ &= V\Sigma^{-1}V^T + V\Sigma^{-1}V_\phi^T(I - V_\phi\Sigma^{-1}V_\phi^T)^{-1}V_\phi\Sigma^{-1}V^T \\ &= \mathcal{L} + \mathcal{L}_{.\phi}(I - \mathcal{L}_{\phi\phi})^{-1}\mathcal{L}_{\phi.} \end{aligned} \tag{5.12}$$

Then the last equality of Eq. (5.10) follows the analytic block-wise matrix inversion formula.

5.2.2 Experiment Settings and Results

To demonstrate the efficiency of our proposed model in the scenario of new user onboarding, we conduct experiments on both the *Flickrgroup* data set and the *MovieLens100K* data set in the following sections. All experiments were performed on a machine equipped with an Intel Core i5 2.3 GHz CPU (Dual Core) and 8 GB memory.

Flickrgroup

We randomly pick 1000 users as the new users then we incrementally add 1000 new users to the network. More specifically, starting with 19,850 users in the network, each time we add one new user to the network and re-train the model *without* the new user's ratings on items.

Note that WRMF method cannot handle new user onboarding as it does not take social relations between users into consideration. So we get a workaround to use the average user profiles of the users who are in the

Table 5.1: Comparison of the models for the new user onboarding problem

Algorithms	handle OCCF	handle new user oboarding	*Flickrgroup*	*MovieLens100K*
PMF	-	-	-	-
WRMF	Y	-	-	-
SoRec	-	Y	-	Y
WRSoRec	Y	Y	Y	-
RWR	Y	Y	Y	Y
GWNMTF	-	Y	-	Y
HNM	Y	Y	Y	Y
LF-HNM	Y	-	-	-
HNM-LF	-	Y	-	Y

same Flickr groups with the new user. We call this workaround *avgWRMF*. Also the RWR method does not require a training phase at all. So we can only compare the time used for re-training with the WRSoRec method, and compare the performance with avgWRMF, WRSoRec and RWR methods. Table 5.1 lists the algorithms that can be evaluated with the one-class *Flickrgroup* and 5-scale *MovieLens100K* data set.

Figure 5.7 illustrates the comparison of the averaged re-training time over 500 new users. We can see that although it needs much longer time to train the initial model for HNM, it takes less than 10 seconds to obtain an update recommendation model for a new user. On the other hand, despite its efficiency to train the initial model, the WRSoRec algorithm requires more than 116 seconds to get the new user's latent profile as well as to update the existing users' profiles.

Figure 5.8 illustrates the P-PS curves for the four testing methods. The mAPS over 1000 new users for avgWRMF, WRSoRec, RWR and HNM is 51.10, 48.02, 39.48 and 35.59, respectively. Although the HNM method performs the best among the four methods, we note that the reported performance on avgWRMF and WRSoRec are poor, as a random score assignment would give mAPS close to 50. It is not surprising that avgWRMF performs poorly since averaging the friends' user profiles may not make sense. However, it is quite surprising that WRSorec, which has shown its power in Section 3.5.4, did not perform well. We note that it is due to the problem of latent factor based methods in the new user onboarding scenario, and we shall discuss more in the coming experiment

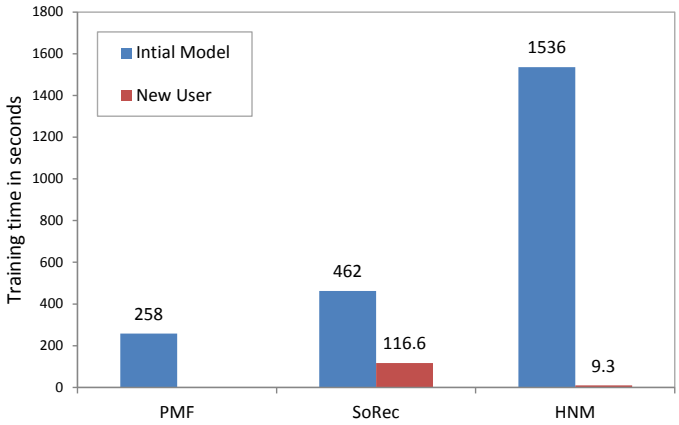


Figure 5.7: Comparison of training time required for both initial model and new user onboarding model update for the *Flickrgroup* data set

on *MovieLens100K*.

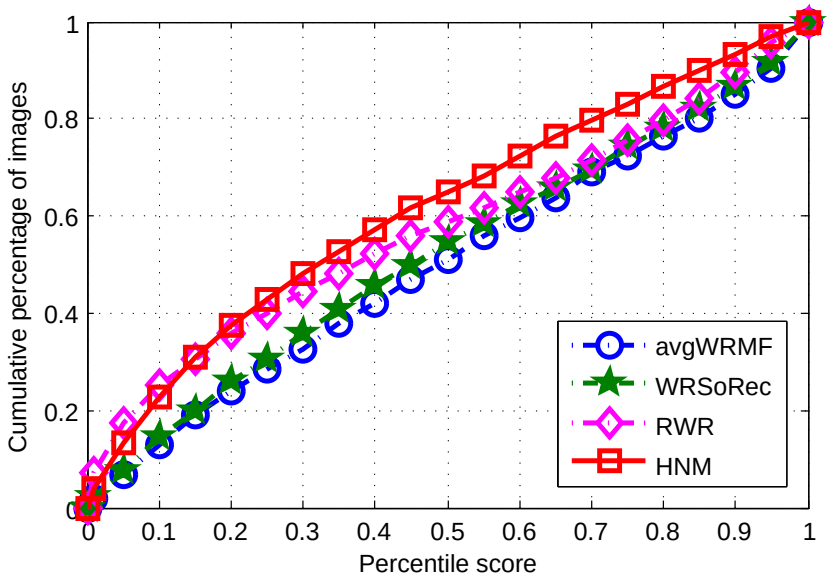


Figure 5.8: P-PS comparison for new user onboarding (*Flickrgroup* data set)

MovieLens100K

Similar to the experiment on *Flickrgroup* data set, we randomly pick 100 users out of 943 users as the new user and remove their ratings to the movies in the training data. The average number of ratings these 100 users given is 99.62.

As listed in Table 5.1, we compare the performance of the five algorithms: SoRec, RWR, GWNMTF, HNM and HNM-LF. The performance results in terms of nDCG are summarized in Table 5.2. First we notice that among the five algorithms, SoRec, GWNMTF and HNM-LF have noticeable low performances compared to the other two algorithms. We note that these three algorithms are all latent factor based algorithm that represents the users by a small number of factors, usually known as the user profile. These factors are learned such that the observed ratings can be reconstructed from the latent factors. Also, there might be some constraints due to the social or other relationships among users when learning the user profiles. Therefore, in the new user scenario where no observed rating is available for the new users, the *only* way to obtain the profiles is based on the social relation constraints with the existing users' profiles. On the other hand, for the RWR and HNM methods, we learn the items' recommendation scores of the new user from *all* available information, even though the new user only connects to his/her friends (or other social relationships). This indeed makes the recommendation score based methods perform much better than the latent factor based methods.

Table 5.2: New user onboarding with *MovieLens1M* data set

Algorithms	nDCG	nDCG1	nDCG5	nDCG10	nDCG20
SoRec	0.4638	0.7459	0.5702	0.4918	0.4709
RWR	0.6337	0.7647	0.6676	0.6468	0.6568
GWNMTF	0.4628	0.6986	0.5475	0.4843	0.4716
HNM	**0.6865**	**0.8025**	**0.6999**	**0.6927**	**0.7016**
HNM-LF	0.4978	0.7934	0.6096	0.5267	0.5103

5.3 Preference Drifting

Although it is common to assume that the users' preferences stay the same in a period of time, it is more reasonable to take the changes of users' preferences over time into account. Similar to the concept drifting that considers the temporal dynamics discovered in the online social streaming, we call this effect the users' *preference drifting*. Figure 5.9 shows such temporal dynamics of users in *MovieLens1M* data set whose preference over

different movie genres drifts. In this figure, X-axis is the movie sequence that the particular user watched order by time and Y-axis indicates the genre(s) that each movie belongs to. It is easy to observe that the user's preference on movie genre drifts over the time. Note that here drifting means that the user's preference on some type(s) of items remains for a while then shifts to other type(s) of items. Preference drifting is a very interesting phenomenon and shall be very useful and important to some scenarios of the recommender system, for example, instant recommendation for online video watching. Yet there is only little literature on studying and exploiting the user preference drifting, such as [45, 46, 47, 48].

5.3.1 Drifting Models

In this section, we discuss how to incorporate the preference drifting in our proposed recommendation model and its effect on the learning algorithm that takes the decay factors on the given preference over time. We first define a time-sensitive weight function, denoted by $p(t; t_0, \xi) = e^{-\xi(t-t_0)}$. This is also referred to as the fading function or time function [45], which regulates the importance of different item preferences for a user over time. This is the exponential decay function, which is used commonly in streaming scenario. The non-negative parameter ξ is the decay rate of the user preferences, which is also related to the inverse of the half-life of the preferences. When $\xi = 0$ there is no decay on the preference, meaning the preference will always be the same over the time. Note that we use the same fading parameter ξ across all preferences.

With the introduction of the fading function $p(t; t_0, \xi)$, we start to define a new time-sensitive recommendation model to address the preference drifting. The major effect is on the edge weight of the user-item matrix where the new time-sensitive weights between user u and item v becomes

$$w(u, v, t) = p(t; t_0, \xi) w(u, v, t_0) \tag{5.13}$$

where $w(u, v, t_0) = w(u, v)$ is the initial weight or rating that the user gives to the item at time t_0. This fading scheme is usually referred to as *fading-to-zero*. However, this fading scheme may be risky as the over discount of the ratings may not truly reflect the drifting of the user's interest. For example,

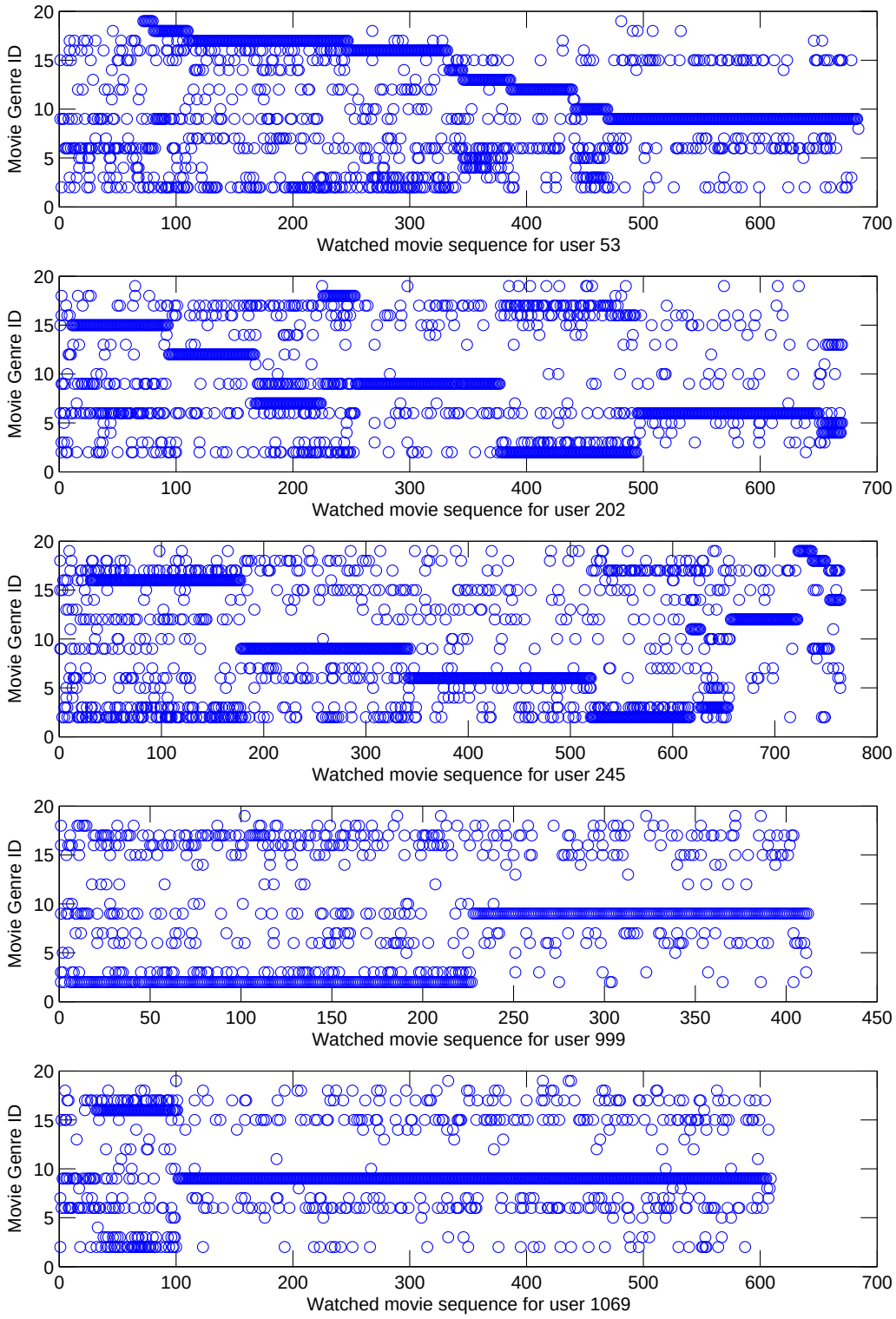


Figure 5.9: Preference drifting in *MovieLens1M* data set

the user's preference on a movie most likely would not change from 5 (very like) to 1 (very dislike) as time goes by. Therefore, we propose a new fading scheme called *fading-to-neutral* as follows:

$$w(u,v,t) = p(t;t_0,\xi)\left(w(u,v,t_0) - w_0(u)\right) + w_0(u) \tag{5.14}$$

where $w_0(u)$ is the neutral rating for the user u which can be the average rating of user u or the average rating of user u in some particular item categories. This means that the fading-to-neutral scheme will make the ratings fade to the user's average rating instead of zero. Figure 5.10 illustrates the difference between these two fading functions.

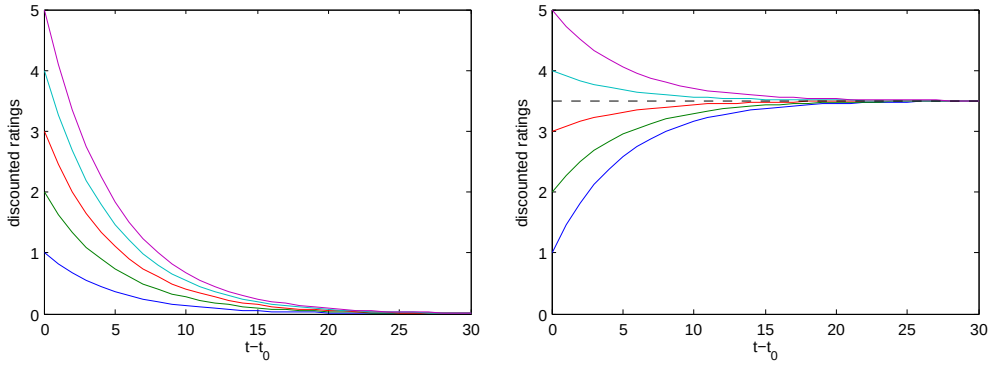
On the other hand, we also make the context-specific target vector y time-sensitive to address the preference drifting. Note that to avoid the y fade to zero, we use a lower-bounded fading function $\bar{p}(t;t_0,\xi,c) = \max\left(c, e^{-\xi(t-t_0)}\right)$ to avoid over discount on the weights. As a result we have

$$y_u(v,t) = \bar{p}(t;t_0,\xi,c)y_u(v,t_0) \tag{5.15}$$

for all the items v that are interested to the user u.

This leads to a new update function for the preference drifting model (t-HNM model)

$$\lambda^{(i)}(f^{(i)} - y^{(i)}(t)) + (I - \Phi^{(i)})f^{(i)} + \sum_{j\neq i}\tilde{\Phi}^{(ij)}(t)f^{(j)} = 0 \tag{5.16}$$



(a) Fading-to-zero (b) Fading-to-neutral

Figure 5.10: Comparison of two fading schemes

5.3.2 Experiment Settings and Results

Data Set and Experiment Setup

It is easier to study the effect of the drifting with the data covering a longer period of time so that users' preference may drift. Since *MovieLens100K* and *Flickrgroup* only cover several months of observations, we conduct the experiment of preference drifting on the *MovieLens1M* data set[1] which is similar to *MovieLens100K* yet covers more users and movies and most importantly, covers a much longer period of observations. Similar to *MovieLens100K*, *MovieLens1M* is also prepared by GroupLens and publicly available. It provides more than one million 5-scaled ratings on 6,883 movies from 6,040 users. Also it contains the demographic information of users (age, gender and occupation) as well as the movie genres in the same way that *MovieLens100K* provides. These ratings were time stamped and cover a period of nearly 35 months or 148 weeks (Apr 2000 to Feb 2003). Figure 5.9 depicts some examples of the user preference drifting on movie genres observed in the *MovieLens1M* data set.

We first partition the data set into observed and unobserved parts based on the rating timestamps. The observed part consists of ratings from Apr 2000 to Dec 2001 (88 weeks) and the unobserved part covers the ratings from Jan 2001 to Feb 2003 (61 weeks). The observed part of ratings is used as the training data for the initial model while the unobserved part is used as the testing data.

Moreover, to not confuse the drifting problem with the *new item* problem, we randomly select one-third of the users (2000 users) as the *pre-observed users* whose ratings are all used as training data, no matter when the ratings are given. The ratings from these pre-observed users are checked to cover all the movies so that no movie is "new" to the model.

The unobserved data is further partitioned into 31 two-week periods of windows which are used sequentially as the testing data. Once the testing window moves to the next period, all the testing periods temporally prior to the testing window become the training data. For each testing window, we report the nDCG as the performance measures.

[1] http://www.grouplens.org/node/73

Compared Algorithms

As there were very few studies on preference drifting in the recommendation tasks, we modify several well-known and state-of-the-art methods so that they can realize the drifting in the model. In particular, we compare our HNM model with the following competing algorithms (see Section 3.5.2 for more details on each algorithm): PMF [10], SoRec [26] and GWNMTF [37]. We note that the original objection function for PMF is as follow:

$$\min_{U,V} \frac{1}{2}\|I \circ (R - g(U^\intercal V))\|_F^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 \tag{5.17}$$

And to realize the drifting in the model, we make the indicating matrix I a time-sensitive weight matrix $I(t)$:

$$I_{ij}(t) = \begin{cases} \bar{p}(t; t_0, \xi, c), & \text{if } R_{ij} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.18}$$

We adopt the same trick to both the SoRec and GWNMTF methods to make them time-sensitive.

Throughout the experiment in this section, we set the fading parameters $\xi \in \{0.1,\ 0.5,\ 0.9\}$ and $c = 0.3$ for the bottom-capped fading function.

Results and Parameter Analysis

Table 5.3 summarizes the performance on preference drifting problem. We use the prefix t- to denote the time-sensitive version of the models. For PMF, SoRec and GWNMTF, the time-sensitive version is simply to adopt Eq. (5.18) in the objective functions. As shown in the table, we first notice that all the time-sensitive versions of algorithms outperform their original non-time-sensitive version in terms of nDCG. Second, our proposed t-HNM achieves the best performance. This, along with the comparison of two different fading scheme used in the t-HNM model (Figure 5.11), would conclude that the fading-to-neutral discounting scheme makes the most contribution for the best performance among all the methods.

On the other hand, Figure 5.12 illustrates the nDCG performance of different settings on the fading parameter ξ. We note that the performance of different settings in PMF, SoRec and GWNMTF models are similar and

their improvements over the original (non-time-sensitive) models are relatively small comparing to that of HNM model. This may indicate the current fading scheme used in these three method could not realize the drifting effect well. Another observation with the same indication is that larger ξ would make possibly over discount on the weights of past ratings. On the other hand, the fading scheme in HNM does outperform the original model that does not realize the drifting effect with a remarkable amount. And larger ξ does not introduce over discount on the weight since the weights do not reduce to zero but to the average weight.

Table 5.3: Average nDCG for 16 testing windows with *MovieLens1M* data set

Algorithms	nDCG	nDCG1	nDCG5	nDCG10	nDCG20
PMF [10]	0.8076	0.8252	0.8086	0.8053	0.8068
t-PMF	0.8252	0.8446	0.8284	0.8235	0.8254
SoRec [26]	0.8091	0.8247	0.8185	0.8112	0.8114
t-SoRec	0.8226	0.8378	0.8329	0.8252	0.8259
GWNMTF [37]	0.8947	0.9045	0.9036	0.8978	0.8992
t-GWNMTF	0.9011	0.9152	0.9061	0.9006	0.9028
BPTF [48]	0.8904	0.9231	0.8977	0.8905	0.8929
HNM	0.8661	0.8742	0.8676	0.8683	0.8628
t-HNM	**0.9117**	**0.9329**	**0.9215**	**0.9186**	**0.9138**

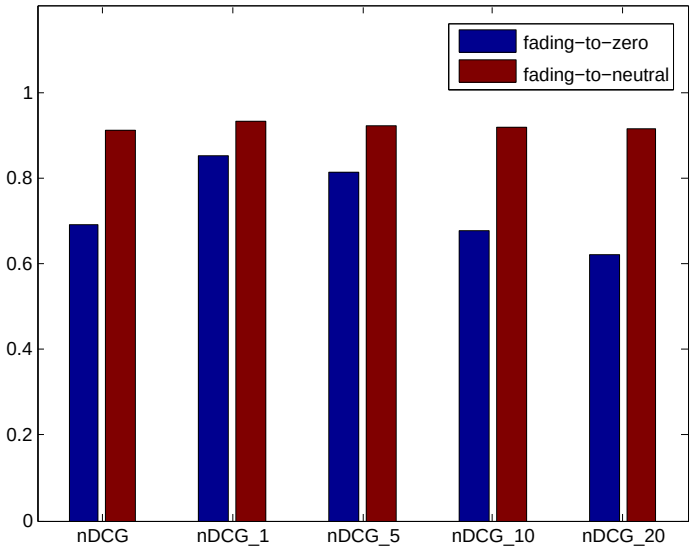


Figure 5.11: nDCG comparison for different fading scheme in t-HNM

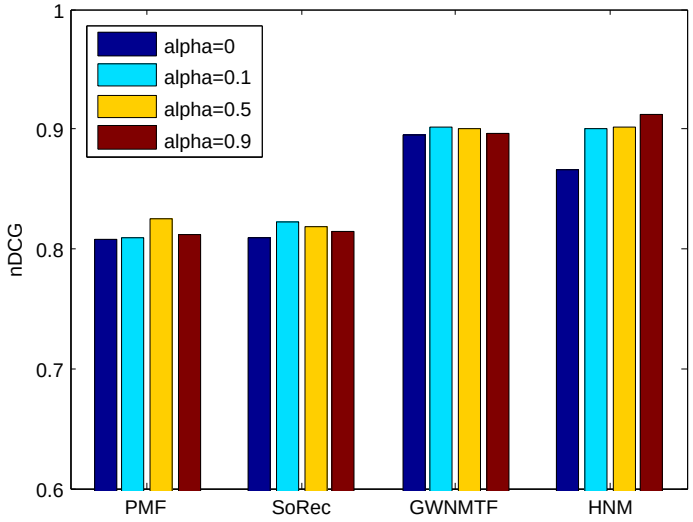


Figure 5.12: nDCG comparison for different fading parameter setting on ξ

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Contributions

In this dissertation, we study recommendation models for heterogeneous social media networks. In addition to the classical recommendation scenario, that is, item recommendations to a specific user, we also look into context-specific recommendations, cold-start recommendations and the preference drifting issue. We summarize the contributions of this work as follows:

- We design a graph regularized recommendation model to use the rich linkage and heterogeneous content information in the recommendation process. In addition, we extend the model by incorporating the latent factors to improve the efficiency and the scalability. To evaluate our approach, we collect a new social media data set for recommendation from Flickr which includes various social, content, preference and contextual information. The experimental results show the effectiveness and efficiency of the proposed approach over the state-of-the-art CF-based or link-based recommender systems.

- We propose a novel recommendation scenario: context-specific recommendations. In particular, with the context-specific recommendation a user or a group of users can obtain recommendations related to specific kinds of content. We show that our proposed approach can be tailored to handle this scenario naturally. In the experiments our approach outperforms the link-based recommendation model in all three different context-specific recommendation scenarios.

- We further discuss the cold-start recommendation for new user onboarding and address the preference drifting issue that commonly

raises in the recommender systems by incorporating a novel drifting paradigm.

- Our approach provides a general framework which can be used for any kind of recommendation in a heterogeneous multimedia network. While this work focuses on image data, the techniques can be effectively used for any kind of heterogeneous social or information network.

6.2 Future Work

In this section, we list several ongoing or potential research directions.

6.2.1 Cross-Source Recommendations

In previous chapters we discussed the utilization of heterogeneous cross-domain data for making effective recommendations. Yet another interesting scenario is to perform recommendations across different social media platforms. This scenario is of great help to deal with the rating-sparsity issue. There are a few studies along this direction, including [49, 50], which assume entity correspondences across sources and use them as constraints to align the embedding of the two sources. While their results seem to be promising, there exist several challenges: (1) there might not be links and correspondences between two sources, and (2) even with user correspondences, user or preference behaviors may differ from one source to another. Due to these challenges, a very recent work [51] assumes no existing correspondences and instead proposes to actively learn the correspondences in conjunction with the regularized CF model. Their model, however, only considers the regularization within each domain and each source. It would be interesting to extend our HNM regularization model to the cross-source scenario.

6.2.2 More Recommendation Scenarios

So far we have discussed a few scenarios in addition to the traditional recommendation scenario. Nevertheless, there are still many interesting

scenarios that are worth exploring.

- *Explainability.* Users oftentimes expect to see the reasons behind the recommendations [52, 53]. There are indeed several positive effects by providing explanations to the users, for instance, it would encourage interaction between users and the recommender systems so as to fix the incorrect prediction and improve the accuracy of the system. Indeed, our proposed HNM model would be able to identify the influential relations that contribute the most to the recommendation score of a particular item, potentially by involving score decomposition and backtracking.

- *Taxonomy.* Many types of social media have manually built taxonomies available, for example, videos and music genre. These taxonomies used to play a very important role in the scenario of new user onboarding. In particular, new users are asked to (optionally) provide an ordered list of genres they favor or disfavor. This information is treated as a prior information to obtain an initial model for those new comers. Although nowadays we can make use of social information to handle the cold start issue, taxonomies still provide a knowledgeable prior information to the recommender system. One possible way to incorporate the taxonomies into the HNM model is to apply the taxonomical matrix to the within-domain matrix. The taxonomical matrix M is a matrix whose entries indicate the weights due to the taxonomy. Then the HNM model can be rewritten as

$$\lambda^{(i)}(f^{(i)} - y^{(i)}) + (I - M \circ \Phi^{(i)})f^{(i)} + \sum_{j \sim i} \Phi^{(ij)} f^{(j)} = 0 \tag{6.1}$$

  where $\circ$ denotes the entry-wise multiplication. Another way to incorporate the taxonomies is to add taxonomical bias to the latent factor model, which has been discussed in [6].

REFERENCES

[1] U. Shardanand and P. Maes, "Social information filtering: Algorithms for automating 'word of mouth'," in *SIGCHI*, 1995.

[2] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *KDD*, 2008.

[3] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS*, 2008.

[4] X. Jin, C. Wang, J. Luo, X. Yu, and J. Han, "Likeminer: A system for mining the power of 'like' in social media networks," in *KDD*, 2011.

[5] G. Adomaviscious and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," in *IEEE TKDE*, 2005.

[6] G. Dror, N. Koenigstein, and Y. Koren, "Web-scale media recommendation systems," in *Proceedings of the IEEE*, 2012.

[7] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Computer Supported Cooperative Work*, 1994.

[8] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *WWW*, 2001.

[9] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola, "Maximum-margin matrix factorization," in *NIPS*, 2005.

[10] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS*, 2007.

[11] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *ICDM*, 2008.

[12] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *ICDM*, 2008.

[13] S. Rendle, C. Freudenthaler, Z. Gantner, and S.-T. Lars, "BPR: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009.

[14] M. Balabanovic and Y. Shoham, "Fab: Content-based collaborative recommendations," in *CACM*, 1997.

[15] M. Pazzani and D. Bilsus, "Content-based recommendation systems," in *Adaptive Web*, 2007.

[16] H. Chen and A. Chen, "A music recommendation system based on music data grouping and user interests," in *CIKM*, 2001.

[17] A. Popescul, L. Ungar, D. Pennock, and S. Lawrence, "Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments," in *UAI*, 2001.

[18] S. Purushotham, Y. Liu, and C.-C. J. Kuo, "Collaborative topic regression with social matrix factorization for recommendation systems," in *ICML*, 2012.

[19] R. W. White, P. Bailey, and L. Chen, "Predicting user interests from contextual information," in *SIGIR*, 2009.

[20] B. Yang, T. Mei, X.-S. Hua, L. Yang, S.-Q. Yang, and M. Li, "Online video recommendation based on multimodal fusion and relevance feedback," in *CIVR*, 2007.

[21] I. Konstas, V. Stathopoulos, and J. M. Jose, "On social networks and collaborative recommendation," in *SIGIR*, 2009.

[22] D. Liu, X.-S. Hua, L. Yang, M. Wang, and H.-J. Zhang, "Tag ranking," in *WWW*, 2009.

[23] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke, "Personalized recommendation in social tagging systems using hierarchical clustering," in *RecSys*, 2008.

[24] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *KDD*, 2008.

[25] Z. Wen and C.-Y. Lin, "On the quality of inferring interests from social neighbors," in *KDD*, 2010.

[26] H. Ma, H. Yang, M. R. Lyu, and I. King, "SoRec: Social recommendation using probabilistic matrix factorization," in *CIKM*, 2008.

[27] E. Spertus, M. Sahami, and O. Buyukkokten, "Evaluating similarity measures: A large-scale study in the orkut social network," in *KDD*, 2005.

[28] Q. Yuan, L. Chen, and S. Zhao, "Factorization vs. regularization: Fusing heterogeneous social relationships in top-n recommendation," in *RecSys*, 2011.

[29] H. Ma, I. King, and M. R. Lyu, "Learning to recommend with social trust ensemble," in *SIGIR*, 2009.

[30] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *RecSys*, 2010.

[31] M. Clements, A. P. de Vries, and M. J. T. Reinders, "Optimizing single term queries using a personalized Markov random walk over the social graph," in *ESAIR Workshop*, 2008.

[32] N. Craswell and M. Szummer, "Random walks on the click graph," in *SIGIR*, 2007.

[33] H. Yildirim and M. S. Krishnamoorthy, "A random walk method for alleviating the sparsity problem in collaborative filtering," in *RecSys*, 2008.

[34] A. Gunawardana and C. Meek, "Tied Boltzmann machines for cold start recommendations," in *RecSys*, 2008.

[35] D. Agarwal and B. C. Chen, "Regression-based latent factor models," in *KDD*, 2009.

[36] C. Wang, R. Raina, D. Fong, D. Zhou, J. Han, and D. Badros, "Learning relevance in heterogeneous social network and its application in online targeting," in *SIGIR*, 2011.

[37] Q. Gu, J. Zhou, and C. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs," in *SDM*, 2010.

[38] X. Yu, X. Ren, Q. Gu, Y. Sun, and J. Han, "Collaborative filtering with entity similarity regularization in heterogeneous information networks," in *IJCAI-HINA*, 2013.

[39] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering," in *RecSys*, 2010.

[40] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *SIGIR*, 2011.

[41] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver, "TFMAP: Optimizing map for top-N context aware recommendation," in *SIGIR*, 2012.

[42] D. Zhou and B. Scholkopf, "A regularization framework for learning from graph data," in *ICML Workshop*, 2004.

[43] X. Zhou, N. Cui, Z. Li, F. Liang, and T. Huang, "Hierarchical Gaussianization for image classification," in *ICCV*, 2009.

[44] S. Zhang, W. Wang, J. Ford, and F. Makedon, "Learning from incomplete ratings using non-negative matrix factorization," in *SDM*, 2006.

[45] Y. Ding and X. Li, "Time weight collaborative filtering," in *CIKM*, 2005.

[46] Y. Koren, "Collaborative filtering with temporal dynamics," in *KDD*, 2009.

[47] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun, "Temporal recommendation on graphs via long- and short-term preference fusion," in *KDD*, 2010.

[48] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with Bayesian probabilistic tensor factorization," in *SDM*, 2010.

[49] B. Mehta and T. Hofmann, "Cross system personalization and collaborative filtering by learning manifold alignments," in *KI*, 2006.

[50] S. J. Pan and Q. Yang, "A servey on transfer learning," in *IEEE Transactions on Knowledge and Data Engineering*, 2010.

[51] L. Zhao, S. J. Pan, E. W. Xiang, E. Zhong, Z. Lu, and Q. Yang, "Active transfer learning for cross-system recommendation," in *AAAI*, 2013.

[52] N. Tintarev and J. Masthoff, "A survey of explanations in recommender system," in *ICDE Workshop on Recommender Systems and Intelligent User Interfaces*, 2007.

[53] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Computer Supported Cooperative Work*, 2000.